Université René Descartes - Paris V
U. F. R. d'Informatique et de Mathématiques Appliquées

Thèse
pour obtenir le grade de
Docteur de l'Université René Descartes

Discipline : Informatique

Contributions à la reconnaissance de l'écriture cursive à l'aide de modèles de Markov cachés

Xavier Dupré

Jury

Georges Stamon	directeur de thèse
Nicole Vincent	
Jacques Labiche	
Emmanuel Augustin	
Françoise Prêteux	rapporteur
Alain Faure	rapporteur

Thèse soutenue le 15 janvier 2004

le 13 juin 2010

Résumé

Ces travaux s'inscrivent dans le cadre de la reconnaissance de l'écriture manuscrite dite hors-ligne. Celle-ci consiste à déchiffrer des mots cursifs présents dans une image. Il n'existe pas encore de solution satisfaisante à ce problème si sa résolution est envisagée de telle manière qu'elle soit valable pour toute sorte de documents. Néanmoins, lorsque ceux-ci sont d'un type précis comme un chèque ou un formulaire, le contexte permet de façonner des contraintes ou des restrictions qui limitent la recherche et autorisent des solutions performantes. Il est plus facile de reconnaître un mot si celui-ci est censé être un nombre dans le cas d'un chèque ou un prénom dans le cas d'un formulaire. Plus généralement, le problème abordé par ces travaux est la reconnaissance de mots cursifs appartenant à un vocabulaire donné.

Le schéma principal des systèmes de reconnaissance de mots cursifs s'articule autour de deux étapes. Il est tout d'abord nécessaire de prétraiter l'image originale du document de manière à en extraire le mot à reconnaître puis à le segmenter en lettres ou morceaux de lettres appelés graphèmes. Ce résultat est ensuite traité à l'aide de modèles mathématiques qui analysent la forme de chaque graphème et leur séquençage. Cette étape probabiliste intègre le plus souvent des modèles de Markov cachés et constitue la reconnaissance proprement dite. Ces modèles sont en effet estimés à partir de grandes bases d'images dont la "connaissance" est résumée dans les coefficients du modèle.

L'étude bibliographique a montré que la recherche dans le domaine de la reconnaissance de l'écriture ne propose pas d'approche très éloignée du formalisme des modèles de Markov cachés. Cette étude s'est prolongée pour aboutir à une description détaillée du processus qui part de l'image pour arriver au résultat afin d'offrir un panorama de la recherche actuelle. Ces travaux ont débouché sur une suite de contributions parvenant à accroître les performances sans pour autant remettre en cause le processus général de la reconnaissance de l'écriture.

Ces améliorations, qu'on peut qualifier de locales, explorent trois thèmes. Elles ont concerné tout d'abord une meilleure prise en compte des erreurs inhérentes au traitement d'images dont la principale tâche se résume à la segmentation d'une image en graphèmes à partir d'heuristiques simples. La prise de décision a ensuite été accélérée afin de pouvoir accroître la complexité du problème de la reconnaissance tout en conservant des temps de traitement raisonnables - de l'ordre de quelques dixièmes de seconde -. Le traitement d'image en lui-même a été retouché afin d'étudier un possible remplacement des heuristiques par des apprentissages sur de grandes bases de données, par l'estimation d'une segmentation en graphèmes traitant les accents de manière séparée, puis lors de la restauration de caractères imparfaits.

L'ensemble de ces travaux est exposé dans un manuscrit qui essaye de proposer un inventaire exhaustif des méthodes utilisées lors de la reconnaissance de l'écriture manuscrite.

Mots-clé

reconnaissance, écriture manuscrite, modèles de Markov cachés, classification, traitement d'images, graphème

Abstract

This work deals with offline handwriting recognition. This task consists in automatically reading cursive words from an image. The existing solutions are not yet good enough if they try to solve any kind of such problems. Nevertheless, when the document obeys a given format such as checks or formularies, the context allows us to build an application which gets satisfactory results. Obviously, it is easier to recognize a word if it is known to be a number written on a check, or a first name written on a formulary. The studied problem in this document is the recognition of words which are known to belong to a given vocabulary.

Most of recognition system include two main steps. First of all, the image source needs to be treated in order to extract the word - which has to be recognized - and to segment it into letters or pieces of letter which are called graphems. Then, this result is processed by mathematic models which analyse the graphem shapes and their sequence. This probabilistic step often includes hidden Markov models and represents the recognition in a strict way. These models are indeed estimated on large databases in order to find their coefficients which can be considered as holding the knowledge.

The bibliographic study showed that the research in the handwriting recognition domain does not offer approaches which are far away from the hidden Markov models' formalism. This study was continued to build a precise description of the whole process from the image to the results. It finally offers a panorama of the actual state of research. The reached objectives are series of contributions which improve the performances without modifying the main scheme of the recognition system.

These local improvements explore three themes. The first one concerns a better way to take into account the unavoidable errors of the graphem segmentation. The second theme is about decision which was accelerated in order to increase models complexity but also in order to keep low processing times (around a few second tenth). The last theme concerns image processing. The aim was to include some training steps instead of many heuristics which are usually used, by training a graphem segmentation, by processing accents in a separate way or by restoring imperfect characters.

Mots-clé

recognition, cursive, hidden Markov models, classification, image processing, graphem

Remerciements

Ces travaux ont été réalisés au sein du laboration des Systèmes Intelligents de Perception (SIP-CRIP5) de l'Université Paris V sous la direction de M. Georges Stamon et en collaboration avec l'entreprise Analyse d'Images et Intelligence Artificielle (A2iA). Je tiens à remercier M. Georges Stamon pour son soutien durant ces trois années. Je remercie l'équipe d'A2iA pour m'avoir accueilli au sein de leur entreprise et pour m'avoir permis de mener à bien ce projet de thèse. Ces travaux poursuivent ceux d'Emmanuel Augustin que je remercie pour toute l'aide qu'il m'aura apporté tout au long de cette aventure. Je remercie Docteur Philippe Gautier, chef du service CIFRE au sein de l'ANRT sans qui ce partenariat n'aurait pas été possible.

Je remercie Marco Cuturi et Jérémie Jakubowicz pour leurs encouragements répétés. Je remercie ma famille et particulièrement mon père pour son soutien inconditionnel, peut-être le seul à percevoir la poésie cachée des formules de mathématiques et à les assembler pour bâtir des cathédrales.

Je joins à ces remerciements Nicole Vincent et Jacques Labiche pour avoir accepté de faire partie du jury et de ce fait de se pencher sur ce manuscrit pour le moins volumineux. Enfin, je tiens à remercier les deux rapporteurs, Françoise Prêteux et Alain Faure, pour leur lecture minutieuse et avisée des nombreuses pages qui suivent.

Repères

La table des matières est détaillée à la fin du document.

Chapitre 1

Introduction

1.1 Avant propos

Une machine capable de lire toute seule sans aucune aide humaine fascine, si bien qu'elle incite à comprendre ses mécanismes internes. Elle fascine tellement qu'elle semble douée d'intelligence artificielle. Ce dernier terme est bien souvent accompagné d'une multitude d'autres comme analyse d'image, modélisation, optimisation... C'est peut-être à ce moment que la machine perd quelque peu de son mystère pour dévoiler ses engrenages complexes qui s'enroulent telle une annonce de défi. C'est peut-être aussi à ce moment qu'on se surprend à vouloir insérer son propre rouage pour croire enfin apercevoir cette machine vous adresser son premier sourire !

1.2 Reconnaissance de l'écriture manuscrite

Quelques articles plus loin, la reconnaissance de l'écriture manuscrite apparaît comme un sujet de recherche toujours vivace et suffisamment vaste pour que très peu d'articles envisagent de décrire un tel système dans sa globalité. De l'image au résultat, deux grandes étapes se succèdent dont la première est une trancription de l'image dans une forme faisant intervenir des espaces vectoriels. Celle-ci fait intervenir de nombreux traitements d'images selon la qualité du document initial. Vient ensuite une modélisation plus formelle qui constitue la reconnaissance proprement dite, on y croise le terme apprentissage cher à l'intelligence artificielle qui équivaut souvent à une optimisation de fonctions aux nombreux coefficients.

Au premier abord, cette seconde partie apparaît comme la plus importante, celle qui, puisqu'elle est apprise, permet de rattraper les imprécisions de la première vouée au traitement d'image. C'est donc a priori vers celle-ci que sont principalement orientées les recherches. Toutefois, depuis quelques années, avec la montée en puissance des ordinateurs, les articles concernant cette modélisation se sont peu à peu taris. Certains ont même conclu que ce domaine avait presque atteint ses limites et que la recherche devait maintenant se diriger vers d'autres cieux comme le traitement d'images auquel la puissance des ordinateurs donne de plus en plus de liberté, ou la prise en compte d'information contextuelle, ou encore l'association de plusieurs systèmes.

Dans un premier temps, le travail a consisté à rassembler par thèmes de nombreuses méthodes utilisées par d'autres chercheurs en ayant pour objectif l'élaboration d'un sytème capable de reconnaître un mot manuscrit depuis l'image jusqu'au résultat. Cette tâche explique la longueur de ce manuscrit. Dans un second temps, chaque brique du système a été construite soit en choisissant l'algorithme le plus performant d'après la littérature, soit en sélectionnant la meilleure méthode à partir de résultats d'expériences, soit

en innovant. Ces travaux ont abouti à l'élaboration d'un système complet de reconnaissance.

1.3 Plan

Il n'est jamais facile de rédiger une synthèse de trois années de travail, d'adopter une logique et de s'y tenir. Les annexes représentent une part importante. Elles pourraient être tout ce qu'il est difficile d'ordonner qui a été jeté pêle-mêle sur le papier. Elles incluent toutefois peu de résultats expérimentaux liés à la reconnaissance de l'écriture, chacune d'elles aborde une modélisation (réseau de neurones, modèles de Markov cachés, ...) ou des méthodes (distance d'édition, plus proches voisins, ...) d'un point de vue mathématique. Ces définitions et théorèmes sont extraits et regroupés par thèmes, ils sont ainsi assemblés de manière plus structurée et plus exhaustive. Ces annexes peuvent ainsi presque se lire indépendamment du reste de ce manuscrit. Comme leur connaissance dépend de chaque lecteur, les premières parties décrivant le travail de cette thèse y feront souvent référence sans détailler ces concepts.

Le premier chapitre reviendra plus particulièrement sur les différentes options de modélisation mathématique de la reconnaissance puis chaque étape du système sera décrite en commençant par le traitement d'image que conclut l'élaboration d'une segmentation en graphèmes. Le troisième chapitre décrira la reconnaissance d'un point de vue mathématique intégrant la description des graphèmes en vecteurs et la reconnaissance. Le quatrième chapitre s'intéressera à la prise de décision, dernière étape avant l'obtention du résultat.

Les annexes terminent le document. Elles regroupent des méthodes issues d'articles et classées par thèmes tels que la squelettisation, les réseaux de neurones, les chaînes de Markov cachées, la classification non supervisée, les distances d'éditions, la recherche de voisins dans un espace métrique quelconque, les n-grammes.

1.4 Notes à propos de ce manuscrit

Ce document est conclu par la liste des tables, des figures et des références, ainsi qu'une table des matières détaillée. Les dernières pages de ce manuscrit forment un index permettant de retrouver notamment les pages où chaque article est cité, les pages où sont écrits définitions, théorèmes et algorithmes. Enfin, cet index contient une rubrique "directions de recherche" qui répertorie chaque page où une possible extension de ces travaux a été envisagée.

Chapitre 2

La reconnaissance de l'écriture : problèmes et solutions existantes

---

Après une brève description des mécanismes sous jacents de la reconnaissance de l'écriture, ce chapitre aborde principalement les modélisations mathématiques utilisées jusqu'à présent dans ce domaine. Celles-ci constituent la partie la plus théorique et sont l'objet de la plus grande part des articles dédiés à ce problème durant la dernière décennie (1990-2000).

2.1 Vue d'ensemble

2.1.1 En ligne, hors ligne

On a coutume de distinguer deux parties dans le domaine de la reconnaissance de l'écriture manuscrite, les reconnaissances *en-ligne* et *hors-ligne*. La reconnaissance dite en ligne s'effectue en même temps que les mots sont écrits, elle concerne les nombreux objets électroniques de poche permettant de saisir du texte sans clavier. La reconnaissance dite hors ligne concerne tout document déjà écrit comme des formulaires, des livres, des chèques.

La reconnaissance en ligne commence à apparaître au travers des annuaires portatifs où la saisie s'effectue en majuscule et lettre par lettre. Elle utilise un stylo et un mécanisme de repérage qui mémorise le tracé. Privée de ces informations, la reconnaissance hors ligne est plus difficile, elle se retrouve souvent cantonnée à des problématiques très précises telles que la lecture d'adresses postales, de montants littéraux de chèques. La gamme de ces problèmes s'étend au fur et à mesure que la puissance des ordinateurs s'accroît.

2.1.2 Styles d'écriture

La difficulté de la reconnaissance est en partie liée au style d'écriture, plus l'écriture est lisible et régulière, plus la résolution est facile. Cette constatation paraît évidente mais si un lecteur humain s'en soucie rarement, les performances obtenues par des logiciels de reconnaissance varient beaucoup avec la clarté des images fournies (lisibilité, bonne résolution de l'image, lignes d'un paragraphe bien espacées, ...). On distingue trois styles d'écriture classés par ordre croissant de difficulté (figure 2.1) bien qu'ils soient parfois emmêlés :

1. l'écriture imprimée,

2. l'écriture manuscrite mais en capitales d'imprimerie ou caractères bâtons,
3. l'écriture manuscrite cursive

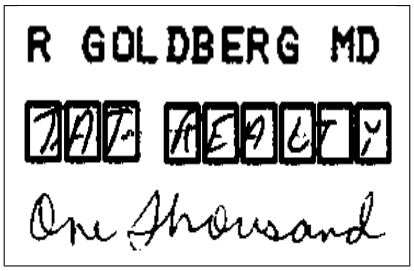


Fig. 2.1: Trois styles d'écriture, imprimé, bâton, cursif.

L'écriture imprimée est un problème pour lequel les solutions existantes sont satisfaisantes. Elles commencent à être accessibles aux particuliers puisqu'elles accompagnent fréquemment les logiciels fournis avec les scanners. Il existe par exemple des logiciels de lecture automatique de carte de visite. L'écriture bâton concerne principalement les formulaires renseignés manuellement comme les feuilles de maladie, les questionnaires à choix multiples. Contrairement à l'écriture imprimée, l'objectif n'est pas de tout décrypter mais au moins d'en traiter automatiquement une bonne partie avec un faible taux d'erreur tandis que les documents mal écrits seront toujours au soin d'un opérateur de saisie. En revanche, les applications traitant la lecture de l'écriture manuscrite sont peu nombreuses. Il s'agit souvent de problèmes précis et réduits comme la reconnaissance du montant littéral d'un chèque, d'une date, quelques champs d'un formulaire.

Par exemple, le ministère de la défense a récemment rendu publique une base de données contenant les fiches de décès de chaque soldat français durant la première guerre mondiale [1]. Toutefois, la législation française ne permet pas de publier un document contenant des informations d'ordre personnel, en particulier médical. Pourtant, le champ contenant la raison du décès est susceptible de contenir ce genre d'information. Le problème consiste donc ici à déterminer si ce champ contient une expression à caractère médical mais sans avoir à la reconnaître. Un traitement informatique a permis de répondre à cette question pour les deux tiers du million de documents avec 0,2% d'erreur.

2.1.3 Problèmes classiques de reconnaissance

La reconnaissance de l'écriture hors ligne recouvre de nombreux problèmes, des plus contraints, reconnaissance d'une lettre parmi une liste prédéfinie, aux moins contraints, reconnaissance d'un paragraphe entier. Le problème de la figure 2.2 illustre la reconnaissance d'un prénom parmi une liste de choix possibles. Ce problème est pour le moment une référence car, pour l'écriture cursive, il est le moins contraint qu'on sache résoudre avec des performances acceptables. Il est désigné plus simplement par l'expression *reconnaissance avec dictionnaire*.

Le terme "acceptable" est volontairement flou, il dépend beaucoup du problème. En ce qui concerne la reconnaissance d'un prénom parmi une liste en contenant environ 2000, pour 100 images, des performances acceptables correspondent à 69 images bien reconnues, 1 image mal reconnue et 30 images rejetées car considérées comme illisibles par le logiciel de reconnaissance. Ces performances sont acceptables parce qu'elles permettent à l'entreprise qui décide de recourir à ces méthodes de réduire le nombre de personnes affectées à la saisie de ces informations, de réduire ses coûts de traitement.

1. Voir le site *http ://www.memoiredeshommes.sga.defense.gouv.fr/* et l'article paru le 13 noveambre 2003 dans *01 Informatique* numéro 1745, page 12.

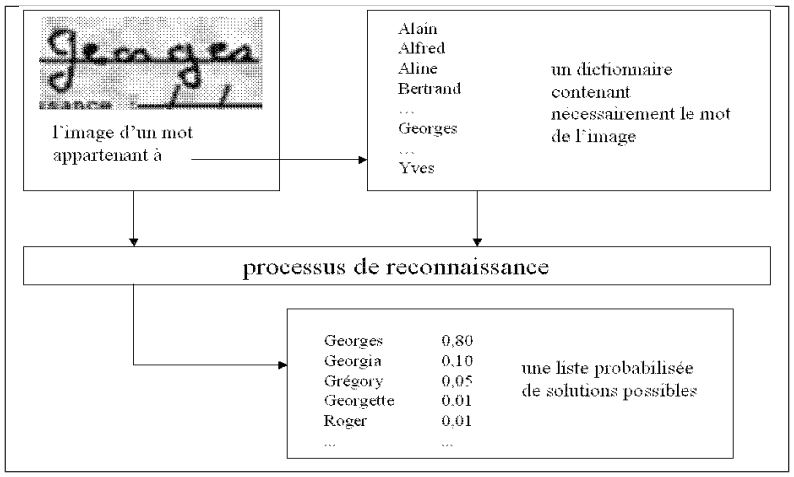


Fig. 2.2: Un problème classique de reconnaissance de l'écriture manuscrite : reconnaissance avec dictionnaire.

Par la suite, les algorithmes proposés, bien qu'utilisés pour l'écriture manuscrite cursive, pourront l'être pour les deux autres styles mais seront vraisemblablement moins performants que des algorithmes spécifiquement développés pour telle ou telle écriture. C'est encore plus vrai pour la reconnaissance en ligne qui utilise des informations supplémentaires relatives au déplacement du crayon.

2.1.4 De l'image au résultat

Le processus de reconnaissance part d'une image et aboutit à une liste de propositions accompagnées d'une probabilité (figure 2.2). Ce long processus peut être découpé en trois parties illustrées par la figure 2.3 :

1. prétraitement de l'image,
2. reconnaissance statistique,
3. décision.

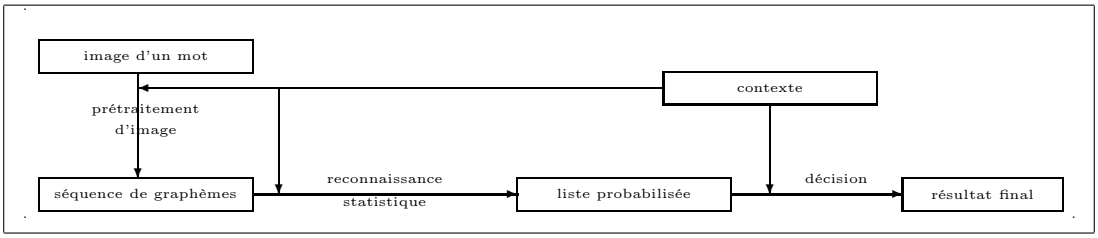


Fig. 2.3: Schéma de l'ensemble du processus de reconnaissance découpé en trois parties : prétraitement de l'image, reconnaissance statistique, décision. La séquence de graphèmes est décrite au paragraphe 3.5. Le terme contexte regroupe toutes les informations inhérentes au problème à résoudre comme la langue, le type de document à traiter, le type d'information à reconnaître (nom, prénom, montant, ...).

La chaîne des prétraitements de l'image comprend essentiellement une segmentation en graphèmes (paragraphe 2.2.1), celle-ci consiste à scinder un problème complexe en plusieurs petits problèmes plus simples. La reconnaissance statistique est centrée autour d'un type de modèle probabiliste adapté à la séquence de graphèmes obtenue à l'étape précédente. Les mieux adaptés sont les modèles de Markov [2] cachés car ils associent séquence et forme des graphèmes. Enfin, l'étape de décision permet d'affiner les résultats de

2. Annexes : voir paragraphe E, page 245

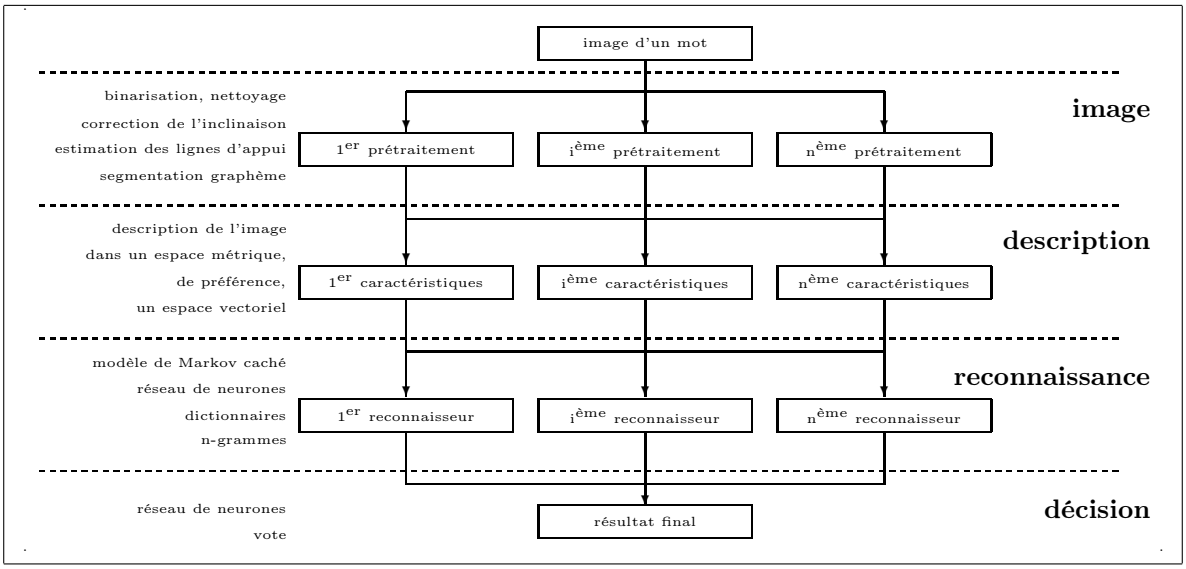


Fig. 2.4: Schéma de l'ensemble du processus de reconnaissance découpé en trois parties principales : prétraitement de l'image, description et reconnaissance statistique, décision. Les systèmes de reconnaissance font de plus intervenir des traitements parallèles, divers reconnaisseurs, mis en compétition afin d'améliorer le résultat final.

la reconnaissance statistique en associant plusieurs reconnaisseurs en tenant compte d'un contexe propre à l'expérience comme le code postal pour la lecture d'une adresse, ce qui permet d'avoir un a priori sur la ville à décrypter. La reconnaisance statistique est fortement dépendante de la langue [3] alors que les prétraitements sont plutôt liés au type de document exploité, enveloppe, chèques, formulaires [4].

Les trois couches du schéma 2.3 sont généralement indépendantes, ce sera le cas pour la majorité des modèles présentés au paragraphe 2.4. Le fait que ces trois étapes s'enchaînent les unes à la suite des autres rend difficile la comparaison de chacune des parties qui composent le système de reconnaissance. Il serait possible de comparer la partie statistique de deux systèmes à partir du moment où les deux autres parties (traitement d'image et décision) sont communes. Les articles publiés présentent rarement un système dans sa globalité et s'intéressent à seulement une de ces trois parties qui sont plus rarement décrites ensemble excepté dans des thèses ou des articles qui en sont issus (voir [Senior1998]). Une exception à cette règle pour l'article [Verma2004], ce dernier décrit sommairement l'ensemble de son système de reconnaissance afin de comparer les performances en reconnaissance de différents types de caractéristiques.

La figure 2.4 illustre de manière exhaustive un tel système. A partir de l'image, plusieurs prétraitements sont possibles, plusieurs segmentations en graphèmes. Chacune d'elles peut à son tour être décrite de manières différentes par des caractéristiques, qui sont à leur tour capables d'être reconnues par des modèles variés. Les résultats sont regroupés puis synthétisés pour aboutir à un seul résultat accompagné d'un taux de confiance. Même si cet ensemble est coûteux à élaborer et à apprendre, il semble que les systèmes de reconnaissance basés sur un prétraitement d'image, un jeu de caractéristiques, un reconnaisseur aient atteint leur limite (voir [Steinherz1999], [Vinciarelli2002] ou paragraphe 2.1.6).

3. La langue dans laquelle les documents ont été écrits est une information importante ne serait-ce que de par le fait que les mots différent d'une langue à l'autre, voire les lettres.

4. Les traitements d'images sont plus dépendants de la qualité des documents (papier, scanner, ...), de leur structuration (formulaire, paragraphe, ...) que du contenu.

2.1.5 Annotation

Par la suite, les termes *annotation*, images *annotées* seront régulièrement employés. L'annotation désigne une information qui accompagne chaque image, celle-ci est relative au contenu et renferme ce que les modèles de reconnaissance devront apprendre. Par exemple, l'annotation la plus simple associée à une image du mot "GEORGES" est le mot "GEORGES" lui-même. Cette information peut être plus étendue et inclure la description d'une segmentation en lettres. En règle générale, l'annotation est construite manuellement et devient la bonne réponse, celle que doit retourner le reconnaisseur s'il a bien appris. Tous les modèles présentés dans ce document sont estimés à partir d'une base d'images annotées de manière à déchiffrer correctement des images non annotées. Selon les modèles, l'annotation requise n'est pas la même. Plus l'annotation est fournie, plus la reconnaissance a de chances d'obtenir de bonnes performances.

2.1.6 Constat et limites

L'article [Vinciarelli2002] brosse le portrait de la reconnaissance de l'écriture cursive et met en lumière les nombreuses étapes résumées dans le paragraphe 2.1.4 qui constituent un tel processus. Les recherches effectuées dans les années précédentes ont abouti aux modèles de Markov cachés qui sont encore à l'heure actuelle la modélisation la mieux adataptée. Leurs performances se sont d'ailleurs concrétisées par la commercialisation de produits offrant de bonnes performances mais dans des domaines précis comme la reconnaissance du montant des chèques ou la lecture d'adresses postales. Ces deux aspects - nombreuses étapes, domaine précis - expliquent pourquoi il est difficile de comparer différents systèmes de reconnaissance. Il est donc impossible de choisir une modélisation plutôt qu'une autre lorsqu'elles n'utilisent pas les mêmes prétraitements. La reconnaissance est aussi améliorée par le contexte de l'expérience qui nuit en même temps à une comparaison fiable de deux systèmes puisqu'ils sont utilisés dans des conditions d'expérience différentes. Dans le cas des chèques, montants littéral et numérique sont mis en correspondances, dans le cas d'une adresse, c'est le cas pour la ville et le code postal.

La reconnaissance de l'écriture manuscrite a donc beaucoup suscité l'intérêt des chercheurs dans les années 1990 à 2000. Les différentes solutions proposées ont difficilement pu être comparées puisque apprises sur des données rarement identiques dans des contextes rarement semblables. Elles proposent malgré tout des solutions exploitables industriellement qui permettent d'envisager la reconnaissance d'un mot manuscrit dans un vocabulaire réduit comme un problème quasiment résolu. Toutefois, ces mêmes modèles ont montré leur limite appliqués à des problèmes moins contraints, où le contexte est moins important. L'article [Steinherz1999] suggère d'ailleurs que la recherche soit plutôt dirigée vers l'amélioration de la décision plutôt que la reconnaissance elle-même.

Ce chapitre décrira sommairement les deux premières parties d'un système de reconnaissance, abordées de manière plus détaillée dans les deux chapitres suivants.

2.2 Les prétraitements d'image

Les articles relatifs à cette partie sont moins nombreux. Les algorithmes utilisés sont plus le résultat d'expériences, d'une succession d'heuristiques, plutôt que le corollaire d'une modélisation théorique de l'image. L'article [Simon1992] s'intéresse à une segmentation d'un mot à partir du squelette de l'image (voir également [Lecolinet1990]). Le résultat souhaité est une segmentation proche des caractères ([Lecolinet1996]).

Le résultat peut être simple et aboutit à des morceaux de petites tailles qui sont plus difficiles à traiter par la suite lors de la reconnaissance statistique. La méthode des fenêtres glissantes (voir [Knerr2001], figure 2.5) fait partie de cette catégorie. L'image d'un mot est simplement découpée en bandelettes verticales sans relation avec les lettres. La reconnaissance est dévolue à des modèles mathématiques complexes.

Le résultat peut être plus travaillé et inclure les résultats de nombreuses expériences sous forme d'heuristiques. La segmentation obtenue est plus proche de celle des lettres et plus instable aussi car ce traitement est un assemblage de règles essaimées lors de squelettisations, de nettoyages ou de segmentations. En contre partie, les modèles de reconnaissance statistique peuvent se limiter à la reconnaissance de caractères même s'il n'est pas encore possible à ce stade de segmenter en lettres de manière parfaite à moins de savoir déjà reconnaître ce qu'est une lettre. Par exemple, d'autres travaux ([Knerr2000]) explorent la possibilité de reconstituer l'information temporelle relative au tracé présente en reconnaissance en ligne à partir de l'image écrite. Le système apprend cette tâche à partir d'une base d'images couplées avec une autre base contenant les déplacements du stylo. La segmentation est conclue par la construction d'une séquence ou d'un graphe de petites images appelées *graphèmes*.

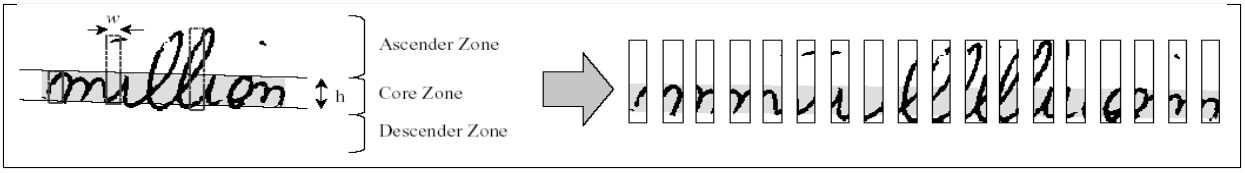


Fig. 2.5: Image extraite de [Knerr2001] illustrant le découpage de l'image d'un mot par des fenêtres glissantes.

2.2.1 Graphèmes

Le sens de lecture de gauche à droite apparente la reconnaissance de l'écriture à la reconnaissance de la parole. L'analogie n'est valable que si l'image en deux dimensions d'un paragraphe (figure 2.6) est décomposée d'abord en lignes (figure 2.7) puis en caractères ou demi-caractères de manière à retrouver cet ordonnancement de gauche à droite comparable au découpage temporel inhérent à la reconnaissance vocale (voir figure 2.8).

Plutôt que d'utiliser des fenêtres glissantes, les premières segmentations élaborées ont tout de suite cherché à isoler les lettres. Ce résultat difficile à obtenir réduit par la suite la taille des modèles de reconnaissance des caractères alors que les fenêtres glissantes aboutissent à des images de caractères décorées de bouts de caractères voisins qui s'apparentent à du bruit. Cette plus grande variabilité mène à des modèles de reconnaissance plus consistants. Cette différence n'est plus importante aujourd'hui mais l'était il y a dix ans lorsque les ordinateurs étaient limités en mémoire.

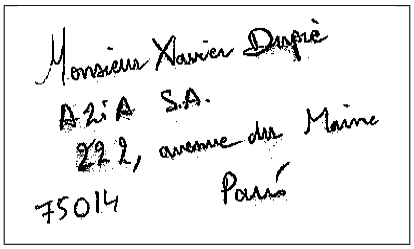


Fig. 2.6: Image d'un paragraphe manuscrit.

Les segmentations en lignes et en mots peuvent être réalisées à partir d'histogrammes (figure 2.7) sommant les pixels noirs de l'image dans une direction parallèle à celle de segmentation. Les pics et les creux de l'histogramme permettent de déterminer sans trop d'erreurs le bon découpage. Comme les histogrammes obtenus sont lissés avant leur exploitation, les résultats sont assez stables à moins que l'image ne contienne des traits parallèles à la direction de segmentation comme un mot souligné. C'est pourquoi l'image est

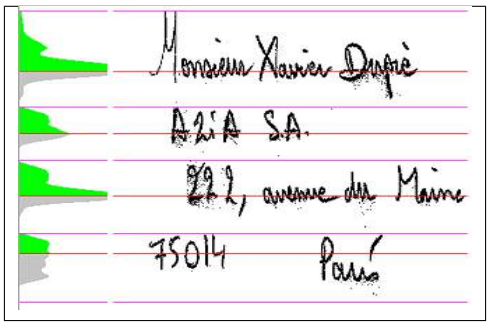


Fig. 2.7: Image des lignes d'un paragraphe figure 2.6 segmentées à partir d'histogramme (partie gauche de l'image).

Fig. 2.8: Graphèmes de la troisième ligne de la figure 2.7.

d'abord nettoyée, débarrassée de pixels isolés et des grands traits horizontaux ou verticaux susceptibles de n'appartenir à aucune lettre. Ce nettoyage est généralement fait à partir d'une squelettisation.

Malheureusement, cette méthode simple n'est pas applicable à une segmentation en caractères. Comme le montre la figure 2.5, les lettres sont souvent penchées et se prêtent mal à un découpage vertical.

Les graphèmes représentent donc une meilleure segmentation puisqu'elle est plus proche des caractères, elle est aussi plus sensible au bruit car elle est parfois basée sur une squelettisation comme l'illustre la figure 2.10 (voir [Lecolinet1990], [Simon1992]). Le squelette est une image plus facile à traiter car il peut être représenté sous forme de graphe. Le découpage est effectué en repérant les ascendants et les descendants sur le squelette, toute forme en "u" est supposée être la frontière entre deux lettres. Par la suite, les règles sont affinées par de nombreuses expériences. Ce prétraitement aboutit à une séquence de petites images appelées *graphèmes* (figures 2.9, 2.10) qu'il faut décrire d'une manière exploitable par des modèles mathématiques de reconnaissance tels que des réseaux de neurones.

Ce prétraitement est constitué d'heuristiques qui le rendent instable et qui sont adaptées au problème de reconnaissance à résoudre, par exemple, la segmentation de chiffres imprimés est différente de celle de chiffres cursifs qui peuvent être attachés.

Pourquoi découper ? Pourquoi ne pas conserver l'image du mot dans son intégralité plutôt que de la segmenter avec une méthode instable ? Deux arguments penchent pourtant en sa faveur :

1. Il est plus facile de décrire des morceaux de lettres plutôt que l'image complète du mot.
2. Dans un problème où les mots sont peu nombreux, il est facile de classer telle ou telle image comme étant tel ou tel mot. Lorsque ce nombre de mots est grand (quelques milliers), la classification devient impossible. Il est préférable de classer des objets plus petits comme les lettres ou des morceaux de lettres.

Un exemple de segmentation sera repris en détail au paragraphe 3.6. Un résultat parfait est difficile à obtenir quelle que soit la méthode si aucune reconnaissance n'est associée à cette segmentation.

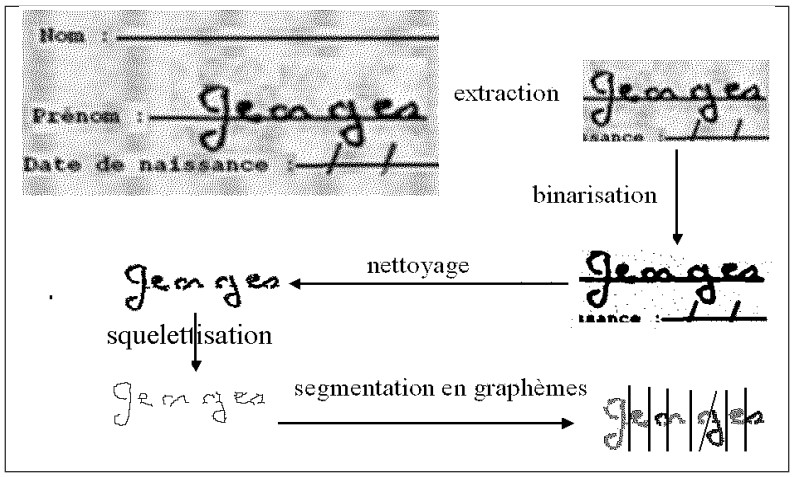


Fig. 2.9: De l'image à la segmentation en graphèmes, le mot est extrait, son image est binarisée, puis nettoyée des bruits résiduels et enfin segmentée en graphèmes souvent en utilisant le squelette. Les nettoyages sont parfois adaptés de manière spéficique à un type de document précis. Ici, le prénom déborde souvent sur l'intitulé *Date de naissance*, les nettoyages sont donc faits en conséquence.

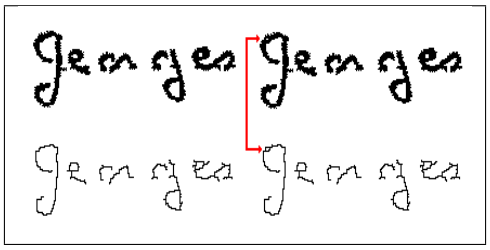


Fig. 2.10: Stabilité de la squelettisation, deux images du même mot dont la seule différence réside dans deux pixels manquant dans le second exemple. Le squelette s'en trouve modifié et la segmentation du premier "g" passe de un graphème à deux.

2.2.2 Segmention explicite-implicite

La litterature (voir [Vinciarelli2002]) distingue parfois deux types de segmentations : implicite et explicite. Lorsque la segmentation en graphèmes a pour objectif le découpage de l'image d'un mot directement en lettres, celle-ci est explicite. La reconnaissance de ce mot est alors réduite à une reconnaissance de caractères. En revanche, lorsque la segmentation en graphèmes découpe cette image en lettres ou en morceaux de lettres, la reconnaissance statistique doit intégrer le fait qu'une lettre est constituée d'un ou plusieurs morceaux. Par conséquent, la segmentation est dite implicite car les lettres n'apparaissent jamais de manière explicite.

L'article [Sayre1973] souligne le paradoxe de la segmentation implicite qui se résume par cette phrase : une lettre ne peut être segmentée avant d'avoir été reconnue et ne peut être reconnue avant d'avoir été segmentée. Par conséquent, les segmentations explicites sont plutôt un compromis, pas assez précises pour définir explicitement des lettres, mais suffisamment pour avoir une association graphème-lettre relativement simple. En règle générale, elles tentent de segmenter l'image d'un mot en morceaux qui sont inclus dans le dessin d'une lettre. Ces segmentations sont souvent regroupées sous le terme *sur-segmentation*.

2.2.3 Caractéristiques

Etant donné que les graphèmes peuvent avoir des tailles variables, l'étape suivante consiste à décrire ces images par un vecteur de caractéristiques de dimension fixe. Ces nombres sont réels et chacun d'eux est censé décrire un aspect de l'image comme :

- la taille du graphème (largeur, hauteur),
- l'aire occupée par les pixels noirs,
- le barycentre des pixels noirs,
- l'inclinaison,
- l'épaisseur moyenne des traits vetircaux et horizontaux, ces chiffres peuvent être calculés sur l'image entière ou une des bandes verticales ou horizontales, l'image est découpée en quatre ou cinq, selon la résolution,
- le nombre moyen de traits verticaux et horizontaux (transition pixel noir - pixel blanc sur une colonne de l'image),
- la position relative du précédent graphème, du suivant.

Ce n'est qu'une description parmi les nombreuses possibles dont un éventail est proposé au paragraphe 4.2. En définitive, ce prétraitement transforme l'image d'un mot en une séquence de vecteurs de dimension fixe ou *mot mathématique*. Si cette dimension est notée D, un mot mathématique est donc :

> **Définition 2.2.1 : mot mathématique**
> On définit un mot mathématique comme une séquence finie de vecteurs de dimension fixe :
>
> $$\text{mot} \longleftrightarrow m \in \left(\mathbb{R}^D\right)^{\mathbb{N}}$$
>
> où D est la dimension de l'espace des caractéristiques.

Le passage d'une séquence d'images à une séquence de vecteurs sera décrit en détail au paragraphe 4.2. La reconnaissance est similaire à un problème de classification. Cette tâche est un problème classique lorsqu'il s'agit d'un espace vectoriel de dimension finie. En revanche, la classification dans l'espace des mots mathématiques (de dimension infinie) est une tâche plus ardue que le paragraphe 2.3 et pour laquelle le paragraphe 2.4 recense des solutions existantes.

2.3 Reconnaissance statistique

2.3.1 Classification en mots

Dans le cas du problème de reconnaissance avec dictionnaire, reconnaître un mot écrit sur une image revient à classer le mot mathématique qui lui correspond dans la classe qui regroupe toutes les écritures différentes de ce même mot (figure 2.11). La reconnaissance est un problème de classification.

Pour chaque classe de mots, un modèle mathématique est construit afin de modéliser l'ensemble des mots mathématiques lui correspondant. Ensuite, pour attribuer la bonne classe à un mot mathématique, tous les modèles de mots sont sollicités et celui qui donne la meilleure réponse est considéré comme la bonne réponse (figure 2.12).

L'avantage majeur de cette organisation est le possible ajout de modèles. Par exemple, en ce qui concerne la reconnaissance du montant littéral des chèques, le passage à l'euro consiste à ajouter le mot *EURO* à l'ensemble des mots possibles pour écrire un montant.

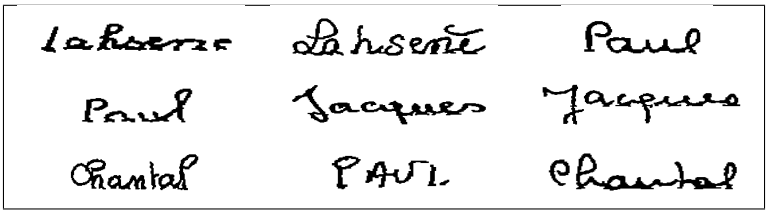
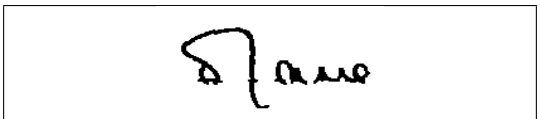


Fig. 2.11: Plusieurs manières d'écrire le même mot.



MARIO	0,782	*MARIA*	0,004
MAIRE	0,108	*FRANC*	0,002
MARIE	0,076	*FRANS*	0,002
MAUD	0,023	*HANS*	0,001
...	...	...	...

Fig. 2.12: Réponse des modèles mathématiques associés aux mots du dictionnaire.

La construction des modèles de mots dépend du problème à résoudre. Toujours dans le cas de la reconnaissance avec dictionnaire, si le dictionnaire est petit (pas plus d'une centaine de mots), un modèle par mot sera construit, c'est-à-dire qu'il aura été appris pour reconnaître ce mot. Dans le cas d'un grand dictionnaire (quelques millieurs de mots), on préférera obtenir ce modèle de mot par l'assemblage de modèles de lettres, qui auront appris à reconnaître spécifiquement les lettres. C'est dans cette optique qu'une bonne segmentation en graphèmes est importante. Si un graphème représente plus d'une lettre, la reconnaissance aura toutes les chances d'être faussée.

2.3.2 Séquence et forme

La définition 2.2.1 définit un mot mathématique comme une séquence de vecteurs, ceci signifie qu'à un mot correspond des mots mathématiques de longueurs différentes et de vecteurs différents mais de dimension fixe. Le paragraphe 2.2.1 a déjà répondu à propos de l'intérêt de conserver un paramètre variable dans la description de l'image d'un mot. Cette description fait donc intervenir deux aspects :

1. une *séquence* de longueur variable,
2. la *forme* des graphèmes car cette séquence est composée de vecteurs de dimension fixe décrivant la *forme* de chacun des graphèmes qui la composent.

Dans les modèles présentés dans ce document, beaucoup traitent ces deux aspects séparément. La forme, le plus simple des deux aspects puisque de dimension fixe, est traitée par un classifieur qui peut être un réseau de neurones, une machine à support vectoriel plus connu sous sa dénomination anglaise *Support Vector Machine* (SVM), un arbre de décision...

La séquence est principalement modélisée par des modèles de Markov cachés. Ceux-ci ont été développés par Baum ([Baum1968], [Baum1972]) et s'inscrivent dans un problème plus général décrit par Dempster ([Dempster1977]) qui expose pour la première fois l'algorithme EM (expectation-maximisation), permettant d'estimer les paramètres d'un modèle prenant en compte des observations cachées. Les modèles de Markov cachés en sont un cas particulier. Pour terminer la liste des articles cités chaque fois, il faut

mentionner une excellente introduction ([Rabiner1986]) dont l'étude est approfondie par [Levinson1983]. Toutefois, ces articles ne présentent que des chaînes de Markov cachées discrètes qu'il faut adapter afin qu'elles prennent en compte l'aspect forme qui, lui, est continu. Un modèle intégrant le couple séquence-forme de manière séparée constitue un modèle appelé *hybride* ou *semi-continu*.

2.3.3 Choix d'une modélisation

Choisir une modélisation mathématique pour un problème impose de faire certaines hypothèses sur les données à représenter (indépendance temporelle, la loi probabiliste qu'elles suivent, ...). Une fois celles-ci converties en un modèle, il peut contenir plus ou moins de degrés de liberté. Plus il en a, plus il peut s'adapter facilement aux données à modéliser et plus il s'adapte difficilement à de nouvelles données.

Les hypothèses doivent donc être les mieux adaptées, elles conditionnent la structure du modèle (modèles de Markov cachés, réseaux de neurones, ...). Il suffit de déterminer le choix du degré de liberté (ou nombre de coefficients) pour le modèle sélectionné. Le choix des hypothèses est toujours le plus important car il détermine les algorithmes d'estimation des paramètres et ceux de leur utilisation. Diverses modélisations sont présentées par la suite.

2.4 Modélisation

2.4.1 Modèles de Markov cachés hybrides et classifieur quelconque

Cette modélisation est utilisée depuis quelques années pour la reconnaissance de textes en écriture romaine et est plus récemment adaptée pour la reconnaissance de textes arabes ([Khorsheed2003]). C'est le modèle le plus simple, à chaque vecteur de caractéristiques est associée une classe de manière à transformer la séquence de vecteurs d'observations continues en séquence d'observations discrètes (figure 2.13). Ce modèle convertit ensuite cette séquence en une probabilité : celle que le modèle émette cette séquence discrète. Ce processus est répété pour chaque modèle de mot, le mot reconnu est celui ayant obtenu la meilleure probabilité d'émission [5].

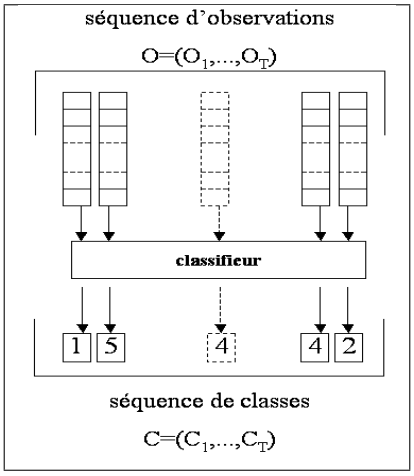


Fig. 2.13: Modèle de Markov caché + classifieur.

Les classifieurs peuvent être de toutes sortes (séparateurs linéaires, centres mobiles, réseau de neurones, réseau de Kohonen, arbre de décision, SVM...). Le choix de ces modèles est influencé par le fait que les

5. Annexes : voir paragraphe E.2, page 247

observations sont annotées (leur classe est connue) ou non. Si elles sont annotées, les graphèmes sont de taille suffisante pour être interprétés (lettres ou parties de lettres), dans ce cas, cette interprétation ou classification donnée est alors apprise par le classifieur [6]. Cette méthode possède l'avantage de fixer le nombre de classes mais aussi l'inconvénient de parfois générer des ambiguïtés car certaines différences apparaissant sur l'image peuvent disparaître dans le vecteur de caractéristiques. De plus, au niveau de l'image, la lettre "u" divisée en deux parties ressemble fortement à deux lettres "i". Si elles ne sont pas annotées, la classification est automatique (centres mobiles par exemple) mais il reste à déterminer le nombre de classes idéal. Cette partie peut faire l'objet d'une étude statistique préalable [7].

Soit un mot mathématique noté $O = (O_1, \ldots, O_T)$ et la séquence des classes d'observations associée à ce mot $C = (C_1, \ldots, C_T)$ (annotation), on définit la probabilité que le modèle M émette la séquence O par :

$$\begin{aligned} \mathbb{P}(O \mid M) &= \sum_{(q_1,\ldots,q_T)} \mathbb{P}(O, q_1, \ldots, q_T \mid M) \\ &= \sum_{(q_1,\ldots,q_T)} \left[\mathbb{P}(q_1 \mid M)\, \mathbb{P}(C_1 \mid q_1, M) \prod_{t=2}^{T} \mathbb{P}(q_t \mid q_{t-1}, M)\, \mathbb{P}(C_t \mid q_t, M) \right] \\ &\quad \text{où } (q_1, \ldots, q_T) \text{ est une séquence d'états du modèle} \end{aligned} \tag{2.1}$$

L'équation (2.1) découle des hypothèses inhérentes aux chaînes de Markov cachées d'ordre un [8]. Cette modélisation fait intervenir deux décisions :

1. La première est prise par le classifieur qui attribue une classe à chaque graphème.
2. La seconde est prise par le module de décision qui décide de lire dans l'image le mot dont le modèle associé est le plus probable.

La seconde décision est incontournable mais la première peut l'être en adoptant une modélisation plus fine comme celle des paragraphes 2.4.2 et 2.4.3.

Ces modèles sont utilisés par [Knerr2001] qui compare leurs résultats à ceux de modèles de Markov cachés discrets. L'intérêt de cet article porte sur l'utilisation de leurs propriétés lors de la segmentation d'une image d'un mot en lettres (figure 2.14). Plusieurs topologies de chaînes de Markov sont exposées accompagnées de l'idée qui a présidé à leur élaboration. Ces modèles sont aussi utilisés dans la thèse [Augustin2001] qui aborde le problème de la recherche de la meilleure topologie de la chaîne de Markov (paragraphe 2.5). Ces modèles font encore l'objet de recherche, [Koerich2002b] présente leur utilisation pour de grands vocabulaires.

2.4.2 Modèles de Markov cachés hybrides et lois gaussiennes

Au lieu de classer les observations afin de les rendre discrètes, celles-ci sont supposées suivre une loi normale, différente pour chaque état du modèle ([Bottou1991]), ou être partagées par plusieurs états à la fois ([Bunke1995]). L'expression de la probabilité d'émission est résumée par les deux équations (2.2) et (2.3) :

6. Cette annotation est en général coûteuse à obtenir puisqu'il faut segmenter manuellement chaque image puis classer chaque morceau ou lettre.
7. Annexes : voir paragraphe H, page 346
8. Annexes : voir paragraphe E.2, page 247

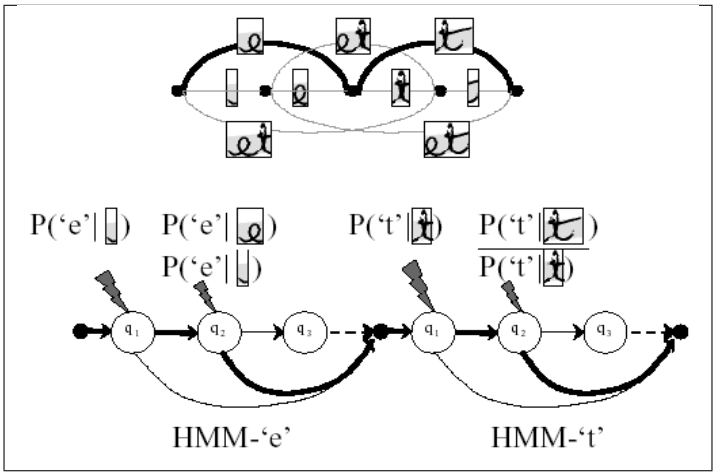


Fig. 2.14: Image extraite de [Knerr2001], elle représente un graphe de segmentation du mot "et", parmi toutes les possibilités, celle en trait gras est la plus probable.

$$\mathbb{P}(O \mid M) = \sum_{(q_1,\ldots,q_T)} \left[\mathbb{P}(q_1 \mid M)\, \mathbb{P}(O_1 \mid q_1, M) \prod_{t=2}^{T} \mathbb{P}(q_t \mid q_{t-1}, M)\, \mathbb{P}(O_t \mid q_t, M) \right] \tag{2.2}$$

$$\text{où } \mathbb{P}(O_t \mid q_t = i, M) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{\det(V_i)}}\, e^{\left(-\frac{1}{2}(O_t-\mu_i)'(V_i)^{-1}(O_t-\mu_i)\right)} \tag{2.3}$$

avec n la dimension de l'espace vectoriel des observations

L'inconvénient de ce modèle est sa gourmandise : pour des vecteurs d'observations de 50 caractéristiques, la matrice carrée V_i contient 2500 coefficients et ce pour chaque état de la chaîne de Markov. Par conséquent, l'utilisation de ces modèles s'accompagne d'hypothèses simplificatrices : V_i est souvent supposée diagonale.

2.4.3 Modèles de Markov cachés hybrides et réseau de neurones

Le classifieur de la figure 2.13 est remplacé par un réseau de neurones illustré par la figure 2.15. Il associe à une séquence d'observations $O = (O_1, \ldots, O_T)$ une séquence de vecteurs de probabilités $V = (V_1, \ldots, V_T)$ où $V_t = \left(V_t^1, \ldots, V_t^L\right)$ et $\sum_{k=1}^{L} V_t^k = 1$. L'expression [9] de la probabilité devient :

$$\mathbb{P}(O \mid M) = \sum_{(q_1,\ldots,q_T)} \mathbb{P}(O, q_1, \ldots, q_T \mid M)$$

$$\mathbb{P}(O \mid M) = \sum_{(q_1,\ldots,q_T)} \left[\mathbb{P}(q_1 \mid M)\, \mathbb{P}(O_1 \mid q_1, M) \prod_{t=2}^{T} \mathbb{P}(q_t \mid q_{t-1}, M)\, \mathbb{P}(O_t \mid q_t, M) \right] \tag{2.4}$$

$$\text{avec } \mathbb{P}(O_t \mid q_t, M) = \sum_{k=1}^{L} \mathbb{P}(O_t \mid C_t = k)\, \mathbb{P}(C_t = k \mid q_t, M)$$

où le vecteur $[\mathbb{P}(O_t \mid C_t = k)]_{1 \leqslant k \leqslant L}$ est retourné par le réseau de neurones (ou $\mathbb{P}(C_t \mid O_t)$)

9. Annexes : voir paragraphe E.5.1, page 282

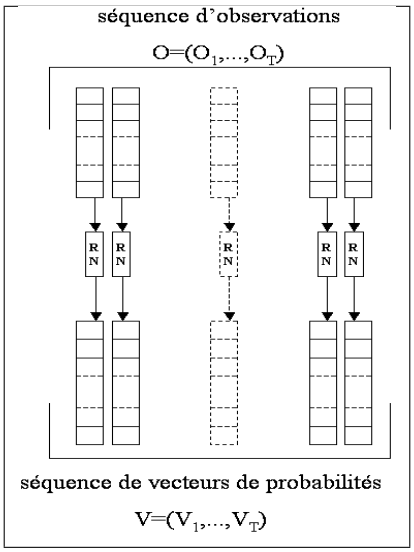


Fig. 2.15: Modèle de Markov caché + réseau de neurones

L'avantage de ce modèle par rapport à celui du paragraphe 2.4.1 est qu'une observation n'appartient plus impérativement à telle ou telle classe, le réseau de neurones ne donne que les probabilités d'appartenance, la décision n'est plus aussi brutale. Ne pas prendre de décision est une expression récurrente dans la reconnaissance de l'écriture, un leitmotiv. Une décision brutale implique un seuillage, c'est-à-dire l'utilisation d'une fonction non dérivable et difficile à apprendre.

L'estimation du modèle est évidemment plus compliquée. L'initialisation du réseau de neurones s'effectue grâce à un classifieur, l'estimation des coefficients de la chaîne de Markov cachée est identique à celle du modèle 2.13. Une fois ces deux étapes effectuées, l'apprentissage du réseau de neurones est supervisé par la chaîne de Markov. L'apprentissage du modèle global résulte d'une alternance entre apprentissage chaîne de Markov et apprentissage réseau de neurones, jusqu'à convergence de l'un et de l'autre.

Les modèles utilisés chez la société A2iA contiennent entre 20000 et 70000 coefficients pour le réseau de neurones, entre 100 et 1000 coefficients pour la chaîne de Markov cachée. Le facteur forme (réseau de neurones) est donc beaucoup plus important que le facteur séquence (chaîne de Markov cachée).

Un apprentissage simultané est envisagé dans [Bengio1992], néanmoins, pour un aussi volumineux système, il est parfois préférable de scinder le problème.

2.4.4 Modèles de Markov cachés hybrides et SVM

Une alternative au réseau de neurones (2.4.3) est apportée par les "Support Vector Machine" (SVM) ou machines à support vectoriel qui ont été proposées par Vapnik ([Vapnik1979], voir aussi [Burges1998]). Leur principe consiste à porter un problème non linéairement séparable dans un espace vectoriel où il le devient. Ces modèles sont proches des réseaux de neurones. Toutefois, leur formalisation permet une plus grande lisibilité des résultats obtenus contrairement aux réseaux de neurones souvent comparés à une boîte noire puisqu'il n'existe pas d'interprétation évidente de leurs coefficients.

2.4.5 Input Output Hidden Markov Models : IOHMM

Le modèle *IOHMM* décrit dans [Bengio1996] est presque une chaîne de Markov cachée. La séquence d'observations $(O_1, \ldots, O_T)$ est expliquée par un processus discret caché $(q_1, \ldots, q_T)$ ou séquence d'états

et un autre processus discret $u = (u_1, \ldots, u_T)$ connu. La probabilité de la séquence d'observations sachant la séquence u selon le modèle M (IOHMM) est donnée par la formule (2.5) (figure 2.16) :

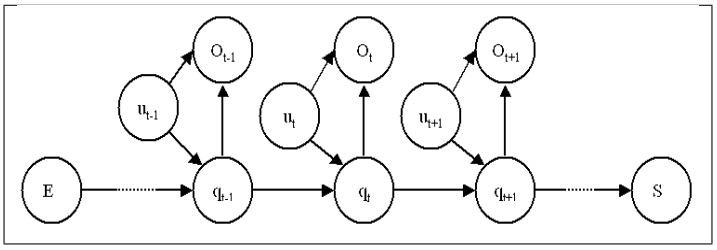


Fig. 2.16: Illustration d'un IOHMM, les flèches indiquent le sens des dépendances.

$$\begin{aligned}\mathbb{P}(O \mid u, M) &= \sum_{(q_1,\ldots,q_T)} \mathbb{P}(O, q_1, \ldots, q_T \mid u, M) \\ &= \sum_{(q_1,\ldots,q_T)} \Bigg[\mathbb{P}(q_1 \mid u_1, M)\, \mathbb{P}(O_1 \mid q_1, u_1, M) \\ &\qquad \prod_{t=2}^{T} \mathbb{P}(q_t \mid u_t, q_{t-1}, M)\, \mathbb{P}(O_t \mid u_t, q_t, M) \Bigg] \end{aligned} \tag{2.5}$$

La séquence u est considérée comme l'entrée du modèle (input) et O sa sortie (output). Ces modèles englobent les chaînes de Markov cachées (il suffit que le processus u soit constant) et permettent aux transitions de dépendre d'un processus connu. Dans ce modèle, la séquence (q_t) dépend de la séquence (u_t), alors que, dans une chaîne de Markov cachée, si on remplace la séquence d'observations (O_t) par (O_t, u_t), les observations (u_t) dépendent de la séquence d'états (q_t). Les IOHMM inversent en partie cette dépendance.

Par rapport aux modèles exposés dans les paragraphes 2.4.1, 2.4.2, 2.4.3, et 2.4.4, les IOHMM proposent une alternative pour la chaîne de Markov cachée, les discussions relatives au choix du classifieur demeurent.

2.4.6 Transducteurs

Les transducteurs sont assez proches des chaînes de Markov cachées puisqu'il est possible de les assimiler à des chaînes de Markov cachées d'ordre $(1, 1)$ (voir [Mohri1996]), c'est-à-dire que l'émission d'une observation dépend à la fois de l'état courant et de l'état précédent, on dit généralement que l'émission se fait sur la transition, ceci se traduit par l'expression (2.6) :

$$\begin{aligned}\mathbb{P}(O \mid M) &= \sum_{(q_1,\ldots,q_T)} \mathbb{P}(O, q_1, \ldots, q_T \mid M) \\ \mathbb{P}(O \mid M) &= \sum_{(q_1,\ldots,q_T)} \left[\mathbb{P}(q_1 \mid M)\, \mathbb{P}(O_1 \mid q_1, M) \prod_{t=2}^{T} \mathbb{P}(q_t \mid q_{t-1}, M)\, \mathbb{P}(O_t \mid q_t, q_{t-1}, M) \right] \end{aligned} \tag{2.6}$$

La correspondance entre transducteurs et chaînes de Markov cachées existe [10] sans toutefois conserver le même nombre de coefficients. Transducteurs et chaînes de Markov cachées sont équivalents. Encore une

10. Annexes : voir paragraphe E.6.12, page 291

fois, par rapport aux modèles exposés dans les paragraphes 2.4.1, 2.4.2, 2.4.3, et 2.4.4, les transducteurs proposent une alternative pour la chaîne de Markov cachée, les discussions relatives au choix du classifieur demeurent.

2.4.7 Class Hidden Model Markov : CHMM

Ces modèles sont décrits par [Krogh1994] ou [Riis1998]. Le modèle CHMM M émet une séquence d'observations $(O_1, \ldots, O_T)$ ainsi qu'une séquence de labels $(l_1, \ldots, l_T)$. Si on note une séquence d'états $(q_1, \ldots, q_T)$, on obtient :

$$\begin{aligned} & \mathbb{P}(O_1, \ldots, O_T, l_1, \ldots, l_T, q_1, \ldots, q_T | M) \\ = \; & \mathbb{P}(q_1|M)\,\mathbb{P}(O_1|q_1, M)\,\mathbb{P}(l_1|q_1, M) \prod_{t=2}^{T} \mathbb{P}(q_t|q_{t-1}, M)\,\mathbb{P}(O_t|q_t, M)\,\mathbb{P}(l_t|q_t, M) \end{aligned} \tag{2.7}$$

Si les labels correspondent aux lettres, alors la reconnaissance d'une séquence d'observations consiste à trouver :

$$(l_1^*, \ldots, l_t^*) = \underset{(l_1,\ldots,l_T)}{\arg\max} \left[\sum_{(q_1,\ldots,q_T)} \mathbb{P}(O_1, \ldots, O_T, l_1, \ldots, l_T, q_1, \ldots, q_T | M) \right] \tag{2.8}$$

L'apprentissage peut être problématique car il demande des séquences d'observations labellisées, c'est-à-dire que chaque observation doit être associée à une lettre. Cette information n'est pas toujours disponible lorsque le nombre de graphèmes diffère de celui des lettres. Si l'annotation de chaque image ne contient que le mot écrit dedans sans aucune indication de position des lettres, ces modèles ne peuvent être estimés.

2.4.8 Generalized Markov Models : GMM

L'apprentissage de modèles de Markov cachés se résume à une optimisation sous contrainte puisque les coefficients de ces modèles doivent être des probabilités. Cette optimisation peut être menée sans contrainte, les modèles obtenus sont alors appelés Generalized Markov Models ([Niles1990], [Balasubramanian1993]). L'apprentissage est plus facile mais ces modèles ne définissent plus une loi de probabilité sur l'espace des observations. Toutefois, Balasubramanian souligne le fait qu'à nombre de coefficients égal, les GMM semblent être de meilleurs classifieurs que les modèles de Markov cachés. Le refrain est ensuite le même puisque par rapport aux modèles exposés dans les paragraphes 2.4.1, 2.4.2, 2.4.3, et 2.4.4, les GMM proposent une alternative pour la chaîne de Markov cachée, les discussions relatives au choix du classifieur demeurent.

2.4.9 Arbres de décision

Un autre modèle de classifieur basé sur un arbre est proposé par [Amit1997]. La méthode décrite allie à la fois construction d'un arbre de classification et sélection de caractéristiques. Les réponses positives de plusieurs filtres matriciels (4x4, 16x16 selon la résolution de l'image) sont recensées et appelées "tag". Il est possible de discriminer deux images de caractères en comparant les distances et les angles formés par les droites reliant les tags (figure 2.17).

La construction d'un tel système passe par un choix adéquat des tags et des relations qui les unissent afin d'obtenir un système de classification robuste et utilisant le moins de règles possible. L'ensemble présenté

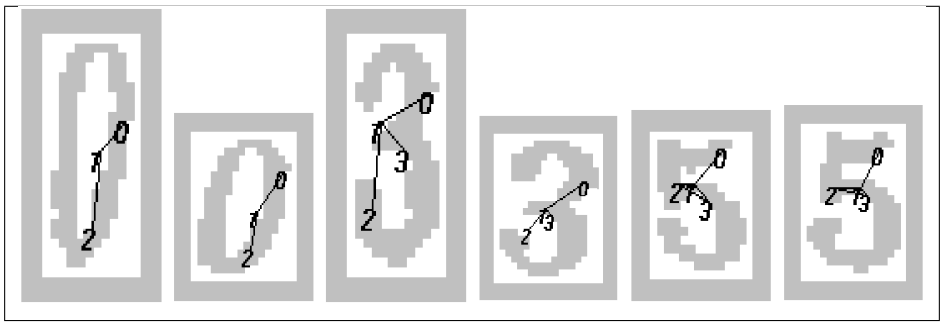


Fig. 2.17: Image extraite de [Amit1997], les tags 0,1,2 sont reconnus dans toutes les images, les images de "0" sont séparées des images de "3" et de "5" par l'absence du tag 3, les images "3" et "5" sont classées en comparant les angles entre les arcs reliant les tags. [Amit1997] propose une méthode permettant de déterminer les relations discriminantes soit l'ensemble minimal de règles permettant de classer chaque image de chiffre. [Amit1997]

par [Amit1997] est un reconnaisseur de caractères mais il pourrait être adapté pour la classification des graphèmes. Dans ce cas, ce système est une alternative aux quatre déjà décrites dans les paragraphes 2.4.1, 2.4.2, 2.4.3, et 2.4.4.

2.4.10 Réseau de neurones incluant des prétraitements d'images

Ce système présenté dans [LeCun1998] est un autre reconnaisseur de caractères basé sur un réseau de neurones. Il diffère des autres par l'incorporation de plusieurs couches de neurones dévolues au traitement d'image (figure 2.18). Son apprentissage est aussi différent des autres puisqu'il est appris à partir d'une base d'images de caractères annotés et de transformations de ces images (inclinaison, rotation, bruitage, ...).

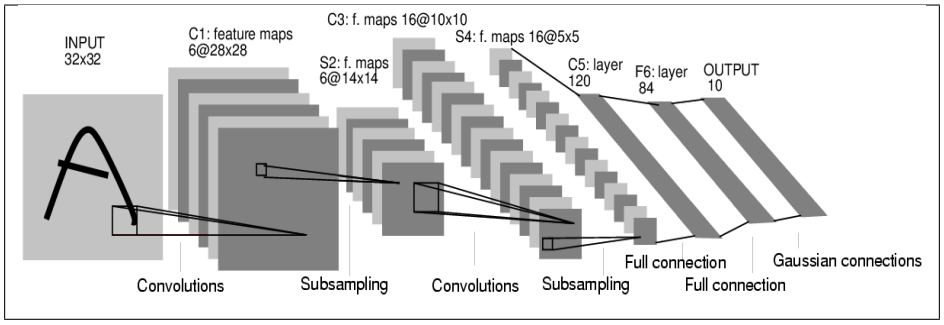


Fig. 2.18: Figure extraite de [LeCun1998] représentant le réseau baptisé LeNet-5, un réseau de neurones à convolution pour la reconnaissance de caractères, tous les plans de la couche C1 partagent les mêmes coefficients, les sorties de cette couche forment un ensemble de caractéristiques pour la couche suivante puisque directement extraites de l'image.

L'ensemble présenté par [LeCun1998] est un reconnaisseur de caractères mais il pourrait être adapté pour la classification des graphèmes. Dans ce cas, ce système est une autre alternative aux cinq déjà décrites dans les paragraphes 2.4.1, 2.4.2, 2.4.3, 2.4.4 et 2.4.9.

2.4.11 Hidden Neural Network : HNN

Ces modèles développés dans [Riis1998] sont construits autour de réseaux de neurones classifieurs [11]. Soit M un HNN possédant N états, à chaque état $q \in \{1, \cdots, N\}$ sont associés deux réseaux de neurones f_q et g_q dont les poids sont respectivement w_q^f et w_q^g. Soit une séquence d'observations $(O_1, \ldots, O_T)$, et $s = (s_1, \ldots, s_T)$ une autre séquence, les probabilités de transition et d'émission du modèle M sont :

$$\begin{aligned} \mathbb{P}(q_t | q_{t-1}, s_{t-1}, M) &= f_{q_{t-1}}\left(w_{q_{t-1}}^f, s_{t-1}\right) \\ \mathbb{P}(O_t | q_t, s_{t-1}, M) &= g_{q_t}\left(w_{q_t}^g, s_{t-1}\right) \end{aligned}$$

On en déduit que, si $(q_1, \ldots, q_T)$ est une séquence d'états (voir figure 2.19), alors :

$$\mathbb{P}(O_1, \ldots, O_T, q_1, \ldots, q_T | s, M) = \mathbb{P}(q_1, s_1, M) \prod_{t=2}^{T} f_{q_{t-1}}\left(w_{q_{t-1}}^f, s_{t-1}\right) g_{q_t}\left(w_{q_t}^g, s_{t-1}\right) \tag{2.9}$$

Les entrées s_t des réseaux des neurones $f_{q_{t-1}}$ et g_{q_t} incluent, comme pour les IOHMM (paragraphe 2.16), des informations complémentaires mais la séquence $(s_1, \ldots, s_T)$ n'est plus discrète.

Ces modèles contiennent trop de coefficients pour pouvoir être appliqués à la reconnaissance de l'écriture car la dimension de l'espace des observations est de quelques dizaines. L'autre inconvénient de ce modèle est de ne pas se prêter facilement à des modifications d'architecture, une suppression d'états entraîne des changements dans la structure de la chaîne de Markov cachée et dans les réseaux de neurones qui lui sont attachés.

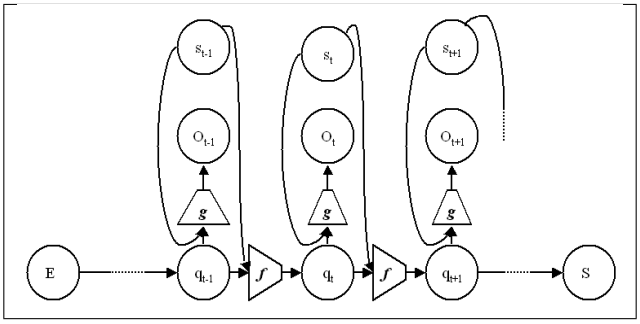


Fig. 2.19: Illustration d'un HNN

Ces modèles sont les plus généraux présentés, ils englobent les IOHMM qui eux-mêmes englobent les modèles de Markov cachés. Ils sont peu utilisés, leur sont préférés les modèles qui suivent utilisant eux-aussi des réseaux de neurones mais sont organisés selon une structure plus simple.

2.4.12 Time Delayed Neural Networks : TDNN

Jusqu'à présent, même s'ils utilisent des réseaux de neurones, les modèles présentés sont explicitement basés sur un squelette incluant le formalisme des chaînes de Markov. Les modèles appelés Time Delay Neural Network ou réseau de neurones à temps retardés s'en passent ([Schenkel1995], [Senior1994], [Senior1998]).

11. Annexes : voir paragraphe C.5, page 217

Le principe de ces modèles illustrés figures 2.20 est d'étiqueter chaque observation selon sa classe d'appartenance (ici des classes de lettres) en tenant compte de ses voisins temporels. Ainsi, un TDNN contenant une couche d'entrée et deux autres couches fonctionnera comme suit :

– La première couche cachée aura pour entrées un vecteur de dimension trois fois celle des observations, la séquence $(O_1, \ldots, O_T)$ deviendra la séquence $\left(S_1^1, \ldots, S_T^1\right)$ où S_i^1 est de dimension D_1 et $S_t^1 = (O_{t-1}, O_t, O_{t+1})$.
– La seconde couche aura pour entrées un vecteur de dimension trois fois D_1 et pour sorties un vecteur de probabilités $(p_1, \ldots, p_T)$ correspondant aux classes de sorties des observations $(O_1, \ldots, O_T)$

Ce genre de réseaux fonctionne comme une fenêtre temporelle locale qui se déplace. Il reste à traiter les effets de bord ou les extrémités de séquences, pour ce faire, on peut supposer que la séquence $(O_1, \ldots, O_T)$ est transformée en la séquence $(O_{-1}, O_0, O_1, \ldots, O_T, O_{T+1}, O_{T+2})$ où $O_{-1} = O_0 = O_{T+1} = O_{T+2} = 0$. Cette manœuvre est répétée pour les deux autres couches.

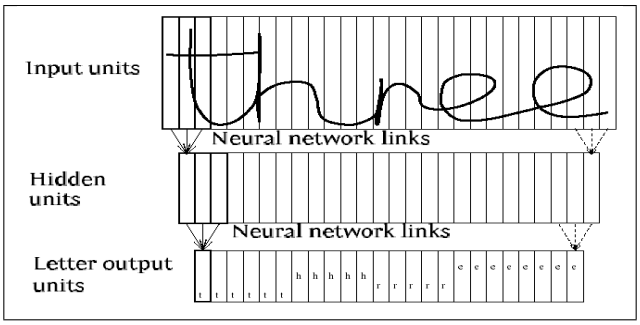


Fig. 2.20: Illustration d'un TDNN, figure extraite de [Senior1994]

Les TDNN associent étroitement forme et séquence, dès la première couche, les graphèmes sont considérés avec leur voisinage et ce jusqu'à la dernière couche qui étiquettera un graphème en tenant compte de ses quatre plus proches voisins, qu'ils soient situés avant ou après dans la séquence, ce que ne font pas les modèles de Markov cachés qui ne prennent en compte qu'une dépendance vers le passé.

L'apprentissage de ces modèles comme celui des CHMM (voir paragraphe 2.4.7) nécessite l'étiquetage de chaque graphème et cette information n'est pas toujours disponible dans les bases de données. C'est pourquoi, ces modèles n'ont pour le moment pas été envisagés seuls mais plutôt comme simples réseaux de neurones dans un modèle hybride comme celui décrit au paragraphe 2.4.3 où la chaîne de Markov cachée fournit l'annotation [12].

2.4.13 Réseau de neurones récurrent

Le TDNN est composé de plusieurs couches qui utilisent les résultats des précédentes estimés sur plusieurs instants consécutifs. Le réseau de neurones récurrent utilise comme entrée l'observation à l'instant t ainsi que des sorties intermédiaires ou finales du réseau de neurones à l'instant $t-1$ (voir figure 2.21, [Senior1994]).

L'article [Senior1998] compare les performances obtenues par un TDNN, un réseau de neurones récurrent, un modèle de Markov caché seul et un modèle de Markov caché hybride associé à un réseau de neurones récurrents. Cette dernière option est la meilleure puisqu'elle permet selon l'auteur d'obtenir un taux d'erreur deux fois moindre par rapport à un réseau de neurones récurrent seul.

12. Annexes : voir paragraphe E.5.3, page 283

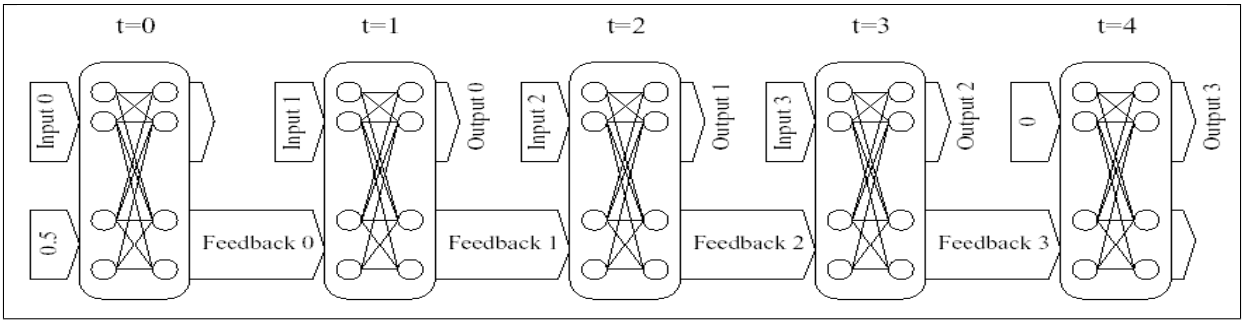


Fig. 2.21: Figure extraite de [Senior1994] illustrant un réseau de neurones récurrant.

2.4.14 Modèle de Markov cachés 2D

Deux versions de ces modèles sont présentées, ils diffèrent des chaînes de Markov cachées simples (1D) car les observations qu'ils modélisent ne sont plus organisées sous forme de séquences indicées temporellement mais sous forme de quadrillage indicé par deux entiers. La figure 2.22 illustre ce principe dans le cas de la reconnaissance d'un visage. De par cette différence, ces modèles 2D ne peuvent plus partir des graphèmes ordonnés sous forme de séquence mais des pixels eux-mêmes ou d'un découpage quadrillé plus grossier.

La première version de ces modèles sont les Pattern Hidden Markov Models ou PHMM, ils sont bien adaptés au traitement d'image et sont utilisés dans la reconnaissance de visages. Leur emploi peut être étendu à la reconnaissance de caractères (voir [Kuo1994]). Il s'agit de modèle à deux niveaux, le premier niveau est constitué de super-états, chacun associé à un modèle de Markov caché simple (voir [Eickeler1998] et figure 2.23).

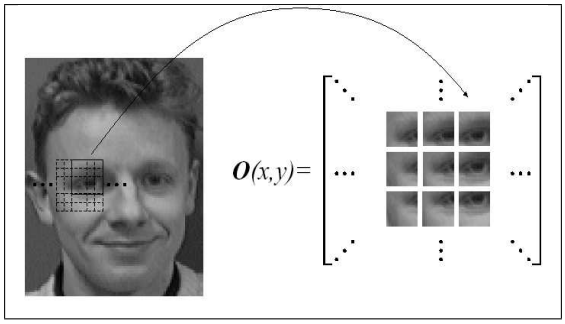


Fig. 2.22: Figure extraite de [Eickeler1998], observations d'un modèle de Markov caché adapté à la reconnaissance de visage

Une seconde version de ces modèles est présentée dans [Saon1997], ce sont les Non-symmetric Half-Plane Hidden Markov models ou NSPH-HMM. Contrairement au modèle PHMM, ils ne comportent qu'un niveau. Ils sont caractérisés par une matrice d'observations $X = (X_{ij})_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n}}$ qui peuvent être les pixels eux-mêmes et une chaîne de Markov cachée $Q = (q_j)_{1 \leqslant j \leqslant n}$. On note λ les paramètres du modèle et Θ_{ij} un voisinage de chaque élément X_{ij} défini comme dans la figure 2.24.

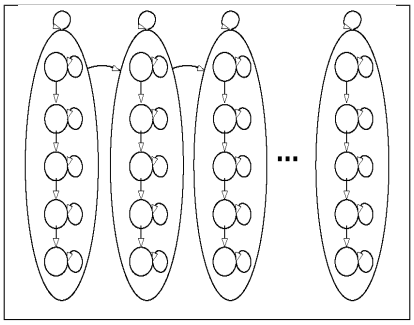


Fig. 2.23: Figure extraite de [Eickeler1998], modèle de Markov caché adapté à la reconnaissance de visage à deux niveaux, le premier niveau (celui contenant les super-états), déplie les sous-modèles selon l'axe des abscisses tandis que le second niveau contient des modèles dont les états se déplient selon l'axe des ordonnées.

$$
\begin{aligned}
P(X \mid \lambda) &= \sum_{Q} \mathbb{P}(X, Q \mid \lambda) \\
&= \sum_{Q} \mathbb{P}(X \mid \lambda)\, \mathbb{P}(Q \mid \lambda) \\
&= \sum_{Q} \prod_{j=1}^{n} \mathbb{P}(q_j \mid q_{j-1}) \prod_{i=1}^{m} \mathbb{P}\left(X_{ij} \mid X_{\Theta_{ij}}, q_j, \lambda\right)
\end{aligned}
$$

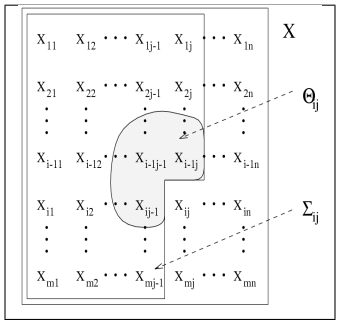


Fig. 2.24: Figure extraite de [Saon1997], voisinage d'un pixel (ou d'une observation) dans le cas d'un modèle NSHP-HMM, le voisinage Θ_{ij} est inclus dans les pixels situés à gauche ou au-dessus d'un pixel Σ_{ij}.

Les PHMM diffèrent des NSPH-HMM par le type de dépendance, dans une seule direction à la fois pour les PHMM, dans les trois quarts du plan pour les NSPH-HMM.

2.4.15 Champs de Markov

Jusqu'à présent, les modèles présentés utilisent des modèles de Markov incluant une dépendance temporelle uniquement tournée vers le passé or, lors du décryptage d'un mot, une lettre mal écrite est en général déchiffrée à l'aide de celles qui l'entourent, autant sur sa droite que sur sa gauche. Il serait donc plus logique que la dépendance temporelle inclut les caractères à gauche et à droite. Cette dépendance n'est plus modélisable par une chaîne de Markov mais par un champ de Markov, modèles souvent utilisés pour la modélisation de textures. Un tel système de reconnaissance utilisant les champs de Markov est décrit dans l'article [Cai2002] et est comparé à d'autres modélisations. Les champs de Markov y obtiennent des performances comparables tout en étant moins gourmands en terme de coefficients.

2.5 Sélection d'architecture

Il est facile de choisir entre deux modèles, il suffit d'estimer leurs paramètres sur une base d'apprentissage comprenant environ 75% des données puis de comparer leurs performances sur une base de test comprenant environ 25% des données ([Bishop1995]). Cependant, elle est mal adaptée au problème de la reconnaissance car l'apprentissage d'un modèle est coûteux (une à plusieurs semaines de calculs). Dans ces conditions, la validation croisée ([Saporta1990]) d'un modèle est superflue. [Bunke1995] envisage quant à lui différentes topologies puis choisit la mieux adaptée à son problème, les architectures envisagées sont encore dessinées par celui qui conçoit le système de reconnaissance et ne sont toutefois pas le résultat d'un algortihme de sélection.

Il est donc préférable d'utiliser des méthodes qui font évoluer l'architecture des modèles au cours de l'apprentissage. Pour une modélisation donnée (une de celles proposées dans les paragraphes 2.4), il faut trouver le bon nombre d'états, le bon nombre de coefficients.

La figure 2.25 représente la modélisation Markovienne de la lettre 'G', comme l'écriture se lit de gauche à droite, celle-ci ne comporte aucun cycle, mais comment savoir que cette lettre s'écrit avec un ou deux graphèmes ? L'objectif des méthodes de sélection d'architecture est d'aboutir au plus petit modèle capable de représenter l'ensemble des données qu'il doit modéliser, ceci revient à trouver la limite entre :

- sous-apprentissage : la modélisation aboutit à de mauvaises performances en reconnaissance,
- sur-apprentissage : la modélisation est excellente sur les données apprises mais s'adapte très mal à de nouvelles données. (voir figure 2.26 et [Bishop1995])

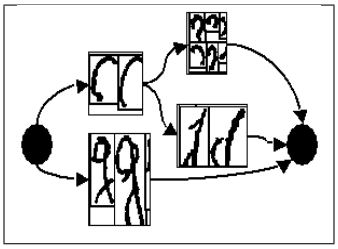


Fig. 2.25: Modélisation de la lettre "G".

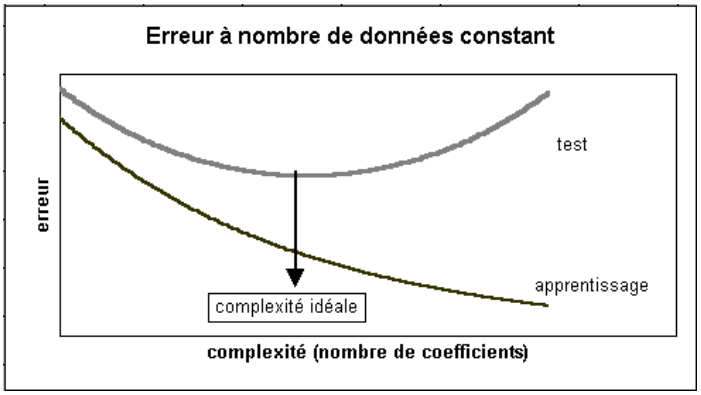


Fig. 2.26: Equilibre entre sous-apprentissage et sur-apprentissage.

La figure 2.26 suggère une définition du modèle optimal : celui qui s'adapte le mieux aux données d'une base de test. Il n'est pas possible d'utiliser cette mesure lors de l'apprentissage auquel cas cette base de test n'en est plus une. La sélection de l'architecture d'un modèle doit donc intervenir uniquement à partir des données d'apprentissage. Les paragraphes qui suivent se proposent de jouer avec la structure des modèles.

2.5.1 Sélection de la structure d'un réseau de neurones

L'article [Cottrel1995] propose une méthode permettant d'estimer la pertinence des coefficients d'un réseau de neurones. Un test statistique ([Saporta1990]) permet de tester la nullité d'un coefficient, de cette manière, il est possible de réduire considérablement le nombre de coefficients et d'améliorer le pouvoir de généralisation du réseau. L'inconvénient de cette méthode est qu'elle nécessite la création d'une matrice carrée de dimension le nombre de coefficients. Pour de grands réseaux ($\sim$ 20000 coefficients), cette méthode amène quelques difficultés d'implémentation au niveau des capacités de stockage en mémoire vive.

Cette sélection utilise des résultats sur les estimateurs du maximum de vraisemblance, ils ne peuvent pas être adaptés tels quels aux modèles de Markov cachés car l'estimation de leurs paramètres dépend d'une optimisation sous contrainte.

2.5.2 Estimateur du nombre d'états d'une chaîne de Markov

L'article [Ziv1992] s'intéresse à un estimateur du nombre d'états d'une source de Markov, cette famille de modèles incluant les chaînes de Markov cachées. On note $\mathcal{P}_j$ l'ensemble des modèles dont le nombre d'états est inférieur ou égal à j, pour une séquence x de longueur n, si $P \in \mathcal{P}_j$, on note $P(x)$ la probabilité de la séquence x sachant le modèle P. Par conséquent, si S_n est l'ensemble des séquences d'états de longueur n, alors :

$$P(x) = \sum_{s \in S_n} \left[\mathbb{P}(x_1, s_1) \prod_{i=2}^{n} \mathbb{P}(x_i, s_i \mid s_{i-1}) \right]$$

On définit N^* un estimateur du nombre d'états du modèle P :

$$N^* = \min \left\{ j \;\middle|\; -\frac{1}{n} \ln \left[\max_{P \in \mathcal{P}_j} P(x) \right] - \frac{1}{n} U_{LZ}(x) < \lambda \right\} \tag{2.10}$$

U_{ZL} est la longueur en bits du code de Lempel-Ziv (LZ) défini dans l'article [Lempel1978]. D'après l'auteur, si N^* est la vraie valeur du nombre d'états, cet estimateur vérifie la condition suivante :

$$\liminf_{n \longrightarrow \infty} \left[-\frac{1}{n} \ln \mathbb{P}(N^* > N) \right] \geqslant \lambda \tag{2.11}$$

n représente la longueur d'une seule séquence mais il peut correspondre également au nombre de séquences finies que le modèle P doit apprendre. Bien que le résultat soit intéressant, le calcul de l'estimateur est encore beaucoup trop coûteux car il nécessite l'estimation de nombreux modèles. De plus, ce résultat intervient pour des chaînes de Markov non cachées.

2.5.3 Réduction du nombre de coefficients d'une chaîne de Markov

Etant donné que les mots mathématiques sont de longueur finie, il est possible de surestimer le nombre d'états optimal nécessaire à un modèle afin de les apprendre. [Augustin2001] propose une méthode dont l'initialisation repose sur un modèle à structure riche (figure 2.27) auquel vont être enlevés états et connexions faibles pour aboutir à une architecture équivalente et réduite.

Les modèles utilisés par [Augustin2001] sont organisés en colonnes d'états, chaque colonne comprend au départ C états où C est le nombre de classes d'observations et chaque état émet une seule classe d'observations (paragraphe 2.4.3). La suppression des états et connexions faibles est conçue comme suit :

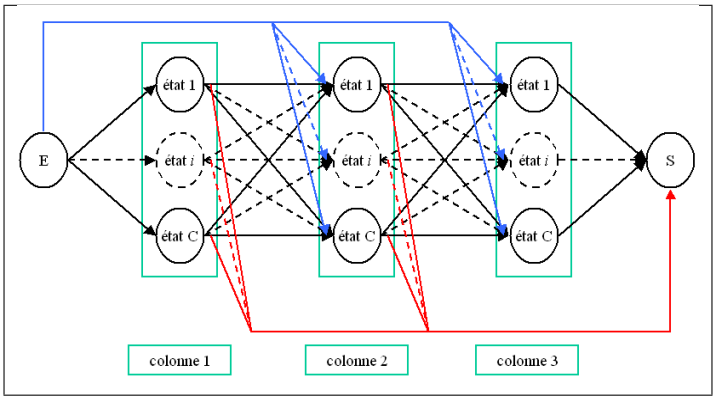


Fig. 2.27: Un modèle à structure riche organisé en colonnes

1. Le modèle est appris grâce aux formules de Baum-Welch (voir [Levinson1983], [Rabiner1986]).
2. Pour chaque séquence :
   (a) Le meilleur chemin d'états est obtenu grâce à l'algorithme de Viterbi [13] (voir [Levinson1983], [Rabiner1986]), il correspond aux écritures les plus probables (voir figure 2.25).
   (b) Pour chaque connexion, on compte le nombre de meilleurs chemins l'empruntant.
   (c) On supprime les connexions trop peu utilisées (en dessous d'un certain seuil), le nombre de connexions supprimées ne peut être supérieur à un petit nombre ($\sim$ 10%), les manières les moins courantes d'écrire telle ou telle lettre sont oubliées.
   (d) Les états n'étant plus connectés sont supprimés également.
3. On retourne à l'étape 1 tant que l'étape 2 supprime des connexions.

La suppression des connexions s'effectue par étape, en théorie, elle devrait s'effectuer une par une, en pratique, pas plus de approximativement 10% sont supprimées. Lorsqu'une connexion marginale est supprimée, une autre qui l'était peut ne plus l'être pour pallier la disparition de la première, c'est pourquoi, les connexions sont supprimées petit à petit. La figure 2.28 donne un exemple de résultat de cette méthode.

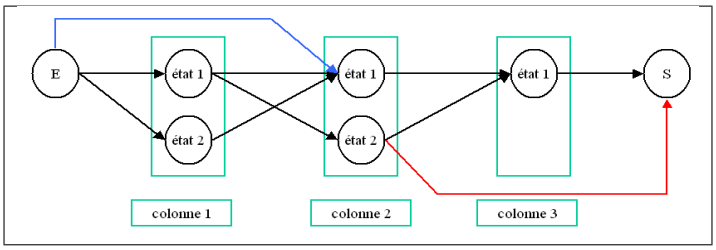


Fig. 2.28: L'architecture sélectionnée par la méthode développée par [Augustin2001]

2.5.4 Bayesian Information Criterium ou BIC

L'article [Bicego2003] adapte aux modèles de Markov cachés une méthode fréquemment utilisée pour la sélection de modèles auto-régressifs appliqués à la prévision de séries temporelles. Cette méthode consiste à pondérer la vraisemblance, toujours croissante lorsque le nombre de coefficients augmente, par un terme décroissant. Le modèle sélectionné doit alors maximiser la vraisemblance pondérée. Cette méthode est

13. Annexes : voir paragraphe E.3, page 260

développée au paragraphe 4.9 qui évoque également les problèmes que soulève son adaptation au cas de la reconnaissance de l'écriture.

2.5.5 Modèles de Markov équivalents

L'article [Kamp1985] propose une méthode pour réduire encore la taille des modèles de Markov cachés, elle permet de détecter le cas des états qui jouent des rôles similaires [14] :

1. les probabilités d'émissions de ces états sont identiques,
2. le regroupement de ces deux états en un seul aboutit à un modèle équivalent, deux modèles M_1 et M_2 sont dits équivalents si quelle que soit la séquence d'observations O, $\mathbb{P}(O \mid M_1) = \mathbb{P}(O \mid M_2)$.

En pratique, l'équivalence de deux modèles est vérifiée uniquement sur la base ayant servi à estimer les modèles.

2.5.6 Assemblage de chaînes de Markov cachées

Le formalisme des modèles de Markov cachés permet également de les assembler de manière simple. En introduisant des états non émetteurs (voir [Lallican1999] [15], un modèle complexe devient la "somme" de modèles plus simples comme sur la figure 2.29.

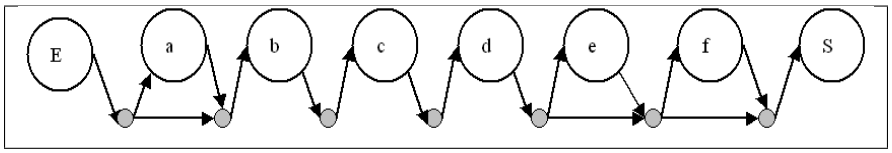


Fig. 2.29: Modèle de mot avec états non émetteurs (états noirs ainsi que les états E (entrée) et S (sortie)), ce modèle est en quelque sorte une imbrication de modèles de Markov cachés notés (a,b,c,d,e,f) et peut représenter les chaînes suivantes : abcdef, abcd, abcde, abcdf, bcdef, bcd, bcde, bcdf.

Les états non émetteurs sont en quelque sorte des aiguillages, ils n'émettent pas d'observations, c'est-à-dire que pour une séquence d'états de longueur T, une séquence d'états associée contiendra T états émetteurs et un nombre quelconque d'états non émetteurs (de zéro à l'infini).

Ce formalisme peut être utilisé pour construire des modèles de mots à partir de modèles de lettres, dans ce cas, vingt-six modèles de Markov cachés seulement modélisent l'ensemble des mots. Cet assemblage peut être adapté aux IOHMM, aux HNN et imaginé comme un jeu de lego qui peut réduire la complexité des modèles en éliminant des connexions redondantes. Ce formalisme ne traite pas à proprement parler de la sélection de modèles mais il propose un moyen intéressant d'assembler les modèles - par exemple des modèles de lettres pour former des modèles de mots - sans accroître le nombre de coefficients.

2.6 Conclusion

La modélisation sous forme de chaînes de Markov cachées est relativement bien adaptée à la reconnaissance de l'écriture manuscrite. Elle permet de traiter aisément des séquences de petites images. Néanmoins, les recherches sur ces modèles s'estompent. Outre l'association de modèles Markov cachés avec différents types de classifieurs, cette modélisation a atteint sa limite sans pour autant être véritablement dépassée

14. Annexes : voir paragraphe F.2.4.4, page 306
15. Annexes : voir paragraphe E.2.6, page 254

par d'autres modélisations. Dorénavant, les chercheurs s'orientent plus vers leur inclusion dans un ensemble plus grand et une meilleure utilisation du contexte (voir [Knerr2001], [Koerich2002b]). C'est dans cette direction que ces travaux seront orientés.

Cette conclusion s'appuie sur l'article [Steinherz1999] qui considère que la reconnaissance de mots cursifs limitée à un lexique réduit est un problème quasiment résolu. La recherche est principalement orientée vers l'optimisation de méthodes déjà existantes, ce dont témoigne la multitude de modèles inspirés du formalisme des modèles de Markov cachés. Toutefois, bien que pour de grands lexiques les résultats obtenus ne soient pas encore satisfaisants, ce même article évoque la possibilité que le problème de la seule reconnaissance d'un mot cursif ait atteint sa limite et que dorénavant, elle doive être dirigée vers des post-traitements pouvant inclure des informations contextuelles ou tout simplement l'utilisation de plusieurs classifieurs. Les HMM et les IOHMM seront les seuls modèles expérimentés. Ce choix peut paraître arbitraire, il est guidé par le fait que la société A2iA utilise déjà ces modèles.

Lors d'une reconnaissance, le processus part de l'image en passant par une modélisation mathématique avant d'arriver au résultat. Les chapitres suivants suivront ce plan, c'est-à-dire la transformation progressive de l'image en un résultat de reconnaissance, rappelant les méthodes déjà développées par d'autres et insérant celles qui sont le fruit de ces travaux. Le chapitre suivant est dédié aux traitements d'images abordés d'une manière plus détaillée que dans ce chapitre. Le chapitre d'après est dédié à la reconnaissance statistique puis viendra le chapitre concernant la décision.

Un dernier chapitre 7 s'intéresse aux nouveaux enjeux, des problèmes pour lesquels des solutions satisfaisantes commencent à faire leur apparition à partir des années 2004-2005.

Chapitre 3

Traitement d'images

Ce chapitre regroupe tous les traitements d'images préalables à l'utilisation de modèles probabilistes qu'on peut scinder en deux ensembles. Le premier corrige les imperfections de l'image comme un bruit importun, un soulignement non désiré, une mauvaise inclinaison. Le second groupe concerne essentiellement la segmentation en graphèmes qui consiste à découper l'image d'un mot cursif en petites images, ceci afin de réduire la complexité des modèles probabilistes utilisés par la suite. La première idée explorée fut d'un apprentissage de cette segmentation. Les résultats insuffisants orientèrent ensuite ces travaux vers la réalisation d'une segmentation à l'aide d'algorithmes plus classiques incluant notamment un traitement dissocié des accents dont la pertinence a été évaluée.

3.1 Préambule

Avant de se lancer dans la reconnaissance à proprement parler, l'image doit être prétraitée de manière à passer d'une information souvent bruitée, toujours de taille variable à une information standardisée. Une série de traitements parfois simples, parfois complexes est d'abord appliquée à l'image avant de la convertir en une séquence d'observations ou mot mathématique, matériau utilisé par les modèles de reconnaissance statistique. L'image d'un mot affronte des traitements tels que l'extraction de la zone à reconnaître, la binarisation, le nettoyage, le redressement de l'inclinaison, la squelettisation, la segmentation en graphèmes, en mots (voir figure 3.1).

Chacune de ces étapes est souvent très rapide et est fréquemment basée sur des heuristiques. L'extraction, la binarisation, la squelettisation [1] sont des traitements communs qui ne seront pas plus détaillés. Le nettoyage est en pratique adapté pour chaque type de problème. Le nettoyage d'un peigne est différent du nettoyage d'une ligne et il n'existe pas encore de méthode générale pour ce type de tâche. Le redressement se réduit à l'estimation de l'inclinaison du texte, une méthode basée sur des histogrammes convient comme celle expliquée au paragraphe 3.3 ou celle du paragraphe 3.4.1. La plupart de ces prétraitements sont décrits sommairement dans [Yanikoglu1998].

L'objectif avoué de cette partie est la conception d'une segmentation en graphèmes, c'est-à-dire le découpage de l'image d'un mot en une succession d'images correspondant à ses lettres ou à des morceaux de ses lettres qui seront utilisés plus tard par des modèles de reconnaissance de l'écriture. C'est un traitement souvent complexe et rarement parfait. Segmentation et reconnaissance sont encore deux étapes distinctes et ceci explique pourquoi ce traitement inclut généralement une multitude de cas particuliers (voir [Lecolinet1991], [Simon1992]). La figure 3.2 résume les faiblesses d'un algorithme de segmentation en

1. Annexes : voir paragraphe B, page 159

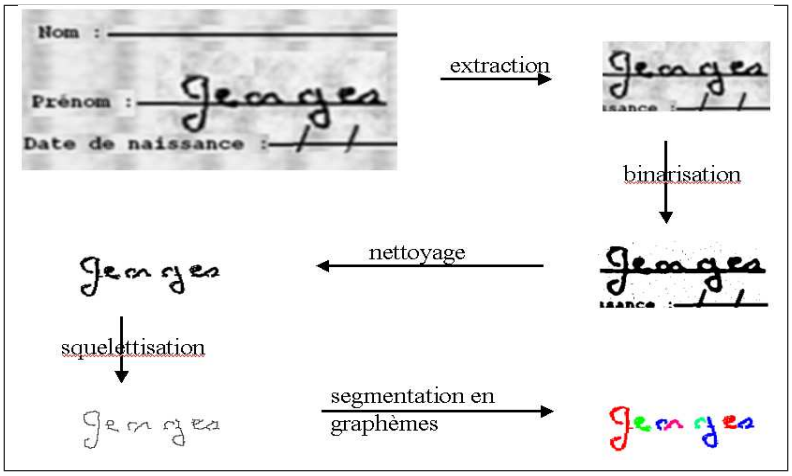


Fig. 3.1: Schéma global des prétraitements d'image.

graphèmes. Ce traitement produit des erreurs quelle que soit la méthode choisie car il est des configurations qui nécessitent la reconnaissance des lettres à segmenter afin d'être tranchées comme la lettre "m" qui se confond avec le couple "rn".

L'algorithme utilisé pour découper les mots de la figure 3.2 segmente mal les couples de lettres à liaison haute comme "oi" contrairement au couple "da" pour lequel, il y a très peu d'erreurs. Il n'est pas évident de juger de l'efficacité d'un algorithme de segmentation en graphèmes, le résultat peut être décevant pour l'œil humain et néanmoins être performant s'il est apparié à des modèles de reconnaissance qui peuvent par exemple modéliser ses erreurs (voir paragraphe 4.7, page 124).

Les paragraphes qui suivent se proposent de décrire différentes méthodes de segmentations (lignes, mots, graphèmes) qui permettront de résoudre le problème de reconnaissance de mots-clé dans un paragraphe manuscrit. Il y aura peu d'évaluation de performances car il est difficile de juger la qualité d'un traitement d'image autrement qu'en observant. La seule sanction est le taux de reconnaissance : combien d'images ont-elles été bien décryptées ? Et dans le cas d'une amélioration des performances, on peut se demander si celles-ci sont dues à une amélioration de la segmentation en graphèmes ou à une meilleure modélisation de cette dernière par des modèles probabilistes.

L'objectif de cette partie n'est donc pas d'améliorer une segmentation graphème existante (celle developpée dans [Baret1991]) mais d'en proposer une autre afin d'obtenir deux chaînes de reconnaissance suffisamment différentes afin que leurs résultats soient si possible corrélés pour des images de bonne qualité mais divergents pour des images de qualité moyenne.

3.2 Apprentissage d'une segmentation

La segmentation en graphèmes présentée par la suite (paragraphe 3.6) s'appuie sur de nombreux seuils fixés "manuellement", ajustés lors de la visualisation du résultat sur quelques images. Ces heuristiques interviennent lors de la segmentation d'une manière qui rend impossible une estimation autre qu'un tâtonnement progressif. Une segmentation pouvant être apprise a l'avantage de pouvoir être modifiée en utilisant les résultats de la reconnaissance. Le second objectif visé est une adaptation plus facile lorsque les documents à traiter changent. De plus, il serait possible d'envisager une boucle alternant les apprentissages de la reconnaissance et de la segmentation automatique.

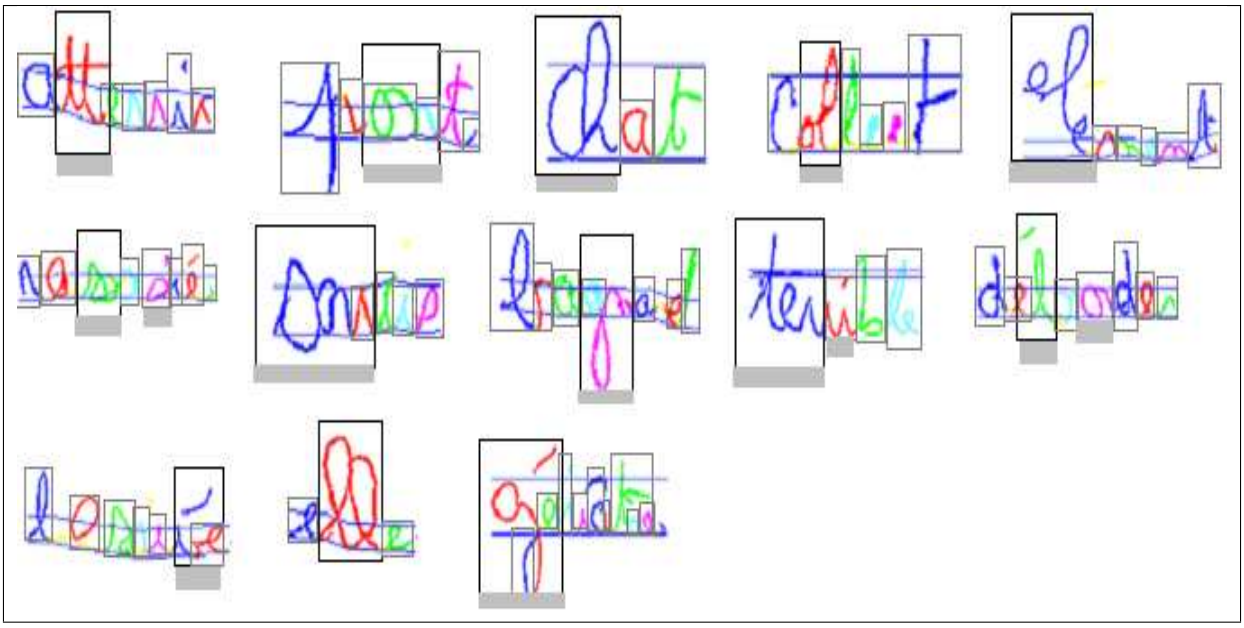


Fig. 3.2: Erreurs de segmentation en graphèmes pour un algorithme (voir [Baret1991]) qui s'appuie essentiellement sur le squelette. Cette opération est erronnée dans environ 10% des cas. Comment segmenter les deux premières lettres du mot "souris" ou "chat" alors que, dans ces deux cas, ce sont presque deux boucles qui possèdent une paroi commune ? Ces configurations sont difficiles à segmenter car les lettres sont souvent écrites de manière enchevêtrée comme les deux "t" consécutifs, les lettres à liaisons hautes (b,o,v,w).

3.2.1 Principe

Cette idée s'appuie sur les diagrammes de Voronoï qui proposent un maillage d'une image (figure 3.3). L'image est d'abord décrite par ses composantes connexes puis réduite à l'état de squelette [2]. Ce squelette est ensuite découpé de manière à ce que les morceaux ainsi formés soient cohérents avec la segmentation désirée. En résumé, aucun des morceaux obtenus ne doit appartenir à deux zones différentes (voir 3.3).

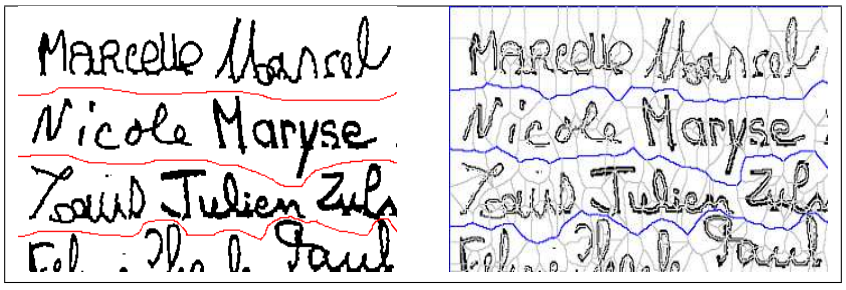


Fig. 3.3: Diagramme de Voronoï utilisé pour une segmentation en lignes : l'image de gauche représente la segmentation à apprendre, les lignes foncées de l'image de droite indiquent les frontières du diagramme de Voronoï correspondant le mieux aux frontières entre les zones de la segmentation désirée.

La segmentation en lignes d'une image telle que celle de la figure 3.3 devient un problème de classification en deux classes : chaque segment du diagramme de Voronoï est une frontière entre deux zones à partager ou ne l'est pas. Comme le montre la figure 3.4, la classification d'un segment peut intégrer des informations relatives aux segments connectés aux deux extrémités ainsi que des caractéristiques sur la forme du texte

2. Annexes : voir paragraphe B, page 159

dans le voisinage de ce segment. L'objectif est la recherche d'une fonction du type :

$$f : S \times S_1^S \times S_2^S \times F_1^S \times F_2^S \longrightarrow [0,1] \tag{3.1}$$

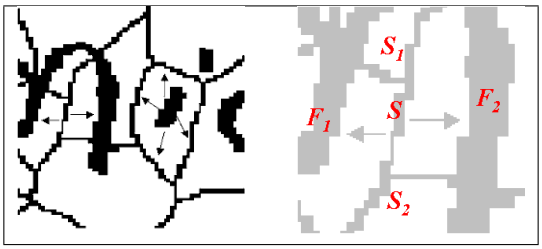


Fig. 3.4: Voisinage d'un segment du diagramme de Voronoï : tout segment est connecté à d'autres segments à ses deux extrémités et il sépare deux zones contenant chacune une petite partie du texte que contient l'image.

S est vecteur caractérisant le segment à classer, S_1^S et S_2^S sont deux vecteurs de même dimension caractérisant les vecteurs connectés à S à chacune de ses deux extrémités, F_1^S et F_2^S sont deux vecteurs caractérisant la forme du contenu des deux zones de textes que S sépare (figure 3.4).

3.2.2 Expérimentations

Dans un premier temps, la fonction f (3.1) a été estimée à l'aide d'un réseau de neurones classifieur [3]. Les vecteurs S, F_1^S, F_2^S contenaient des informations relatives à la longueur du segment, sa courbure, son inclinaison, la distance du segment au texte. Les vecteurs S_1^S et S_2^S contenaient des moyennes des mêmes informations. L'estimation de la fonction f a conduit au résultat figure 3.5 avec un pourcentage de bonne classification proche de 95%.

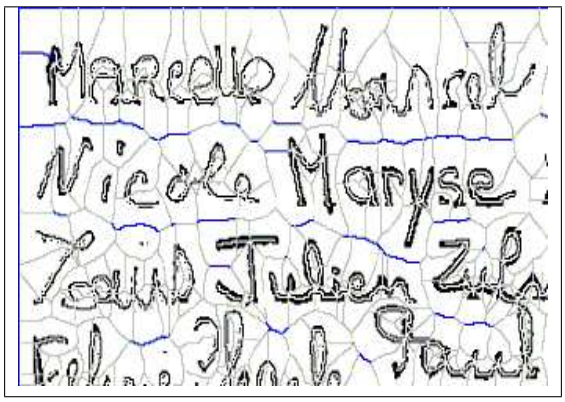


Fig. 3.5: Diagramme de Voronoï utilisé pour une segmentation en lignes : le résultat laisse apparaître des lignes en pointillé. Dans 95% des cas, les segments de Voronoï sont bien classés.

Même si le pourcentage d'erreur est faible, il mène à l'apparition de lignes "trouées" qui suggère soit l'abandon de la méthode, soit son perfectionnement selon deux directions qui sont la création d'un processus itératif permettant de faire évoluer la probabilité d'un segment en fonction de ses voisins et un post-traitement dont l'objectif est l'élimination des "trous". La première direction passe par la construction d'une suite (p_t) pour chaque segment de telle sorte que :

3. Annexes : voir paragraphe C.1.5, page 196

$$p_0^S = f\left(S, S_1^S, S_2^S, F_1^S, F_2^S\right)$$
$$\forall t \geqslant 0,\ p_{t+1}^S = g\left(p_t^S, S, p_t^{S_1}, S_1^S, p_t^{S_2}, S_2^S, F_1^S, F_2^S\right)$$

Le processus s'arrête lorsque la suite $\left(p_t^S\right)_{t\geqslant 0}$ converge pour chaque segment S. Il reste à estimer la fonction g. Le nombre d'itérations nécessaires à la convergence d'un tel système demeure inconnu. La seconde direction correspond en quelque sorte au nettoyage des résultats retournés par la fonction f (ou son prolongement g), les lignes presque achevées sont complétées, les bouts de lignes ne menant à rien sont effacées.

Cette méthode s'appuie sur un diagramme de Voronoï qui peut s'avérer instable lorsque l'image est de mauvaise qualité, lorsque quelques pixels égarés créent des régions artificielles. Les diagrammes de Voronoï flous ([Zhao2000]) seraient peut-être une alternative à ce problème. De plus, la convergence de l'ensemble n'est pas assurée et peut déboucher sur des temps de traitements longs inconvenants pour des applications telles que la reconnaissance de l'écriture. Aucun des deux prolongements n'a été étudié.

3.2.3 Extension au problème de nettoyage

Le nettoyage est un problème dual du précédent puisqu'au lieu de classer les segments du diagramme de Voronoï, il suffit de classer les zones délimitées par ce diagramme en deux classes : zone à nettoyer ou non. L'avantage du diagramme de Voronoï est de proposer un voisinage (figure 3.4) pour chaque petite région. Une application pratique est la suppression d'une ligne qui sert de guide pour l'écriture comme celle montrée figure 3.1. L'intérêt de la méthode réside toujours dans son apprentissage et son inconvénient dans la forte sensibilité du diagramme de Voronoï aux ruptures de connexité.

3.2.4 Diagramme de Kohonen

Outre le fait que le diagramme de Voronoï est très sensible au bruit, pour une région donnée, le nombre de voisins est très variable, il est alors nécessaire de résumer l'information contenue par ce voisinage. On utilise une carte de Kohonen dont la structure est celle d'un quadrillage. Les pixels noirs attirent les neurones qui étirent les arêtes qui les relient comme le montre la figure 3.6a. Les arêtes les plus grandes forment des ponts entre deux régions, un simple seuillage (figure 3.6b) permet presque d'isoler les mots. L'avantage de cette nouvelle structure est son voisinage de taille fixe, quelle que soit la déformation du treillis de Kohonen, chaque neurone conservera quatre voisins, il est alors possible d'utiliser des algorithmes (relaxation probabiliste, champs de Markov) permettant de classer les arêtes en deux catégories : arête à l'intérieur d'une région, arête reliant deux régions à segmenter.

L'inconvénient de cette méthode réside dans l'obtention du treillis final de Kohonen, la convergence est gourmande en temps de calcul pour de grandes images. C'est pour cela que cette idée n'a pas été poursuivie. En revanche, ce temps de calcul devient acceptable si la dimension de l'image est celle d'un mot, cette méthode pourrait donc être utilisée pour apprendre une segmentation en graphèmes.

Cet apprentissage nécessite malgré tout de nombreuses images pour lesquelles la segmentation en graphèmes doit être connue. L'obtention d'une telle base de données peut être manuelle mais ce travail est long ou effectué à partir d'un système de reconnaissance déjà existant mais contenant des erreurs. Les mots les mieux reconnus sont alors découpés en graphèmes ou caractères selon l'usage désiré puis serviront d'apprentissage. Cette direction n'a pour le moment pas été envisagée, une autre permettant de modéliser des erreurs de segmentation en graphèmes lui a été préférée dans un premier temps (voir paragraphe 4.7, page 124). Cette modélisation permet d'ailleurs une meilleure appréciation de la segmentation en graphèmes.

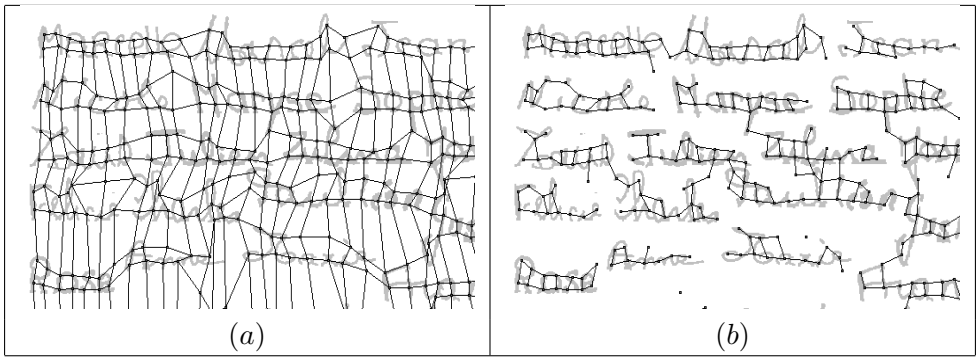


Fig. 3.6: Treillis de Kohonen appliqué à la segmentation en ligne, l'image (a) présente le résultat après convergence des neurones, l'image (b) représente le même treillis dont les arêtes les plus grandes ont été ôtées. Il reste dans le meilleur des cas des assemblages connexes recouvrant l'image d'un des mots.

3.3 Segmentation en lignes

Lors de la scannerisation d'un document, il peut arriver que celui-ci soit incliné (figure 3.7). La première étape consiste donc à redresser une image de telle sorte que les lignes qui la composent soient horizontales. Ce paragraphe aborde diverses solutions existantes et résume les résultats énoncés dans [Dupré2000].

3.3.1 Redressement de l'inclinaison de l'image

De nombreuses méthodes sont utilisées pour détecter l'inclinaison des lignes, leurs robustesses variant avec la difficulté du problème. L'article [Cao2003] par exemple propose une méthode plus adaptée aux textes imprimés. Les composantes connexes (des lettres principalement) sont toutes décrites par un point situé au milieu du bord inférieur de leurs boîtes englobantes. Par la suite, ces points sont regroupés et classés en lignes. Une régression linéaire sur chacune des lignes termine l'estimation de l'inclinaison de l'image. Une autre méthode présentée dans [Lu2003] utilise des chaînes de plus proches voisins (ou nearest neighbors chains), celles-ci sont constituées par l'appariement de voisins. L'inclinaison est mesurée sur chacune des chaînes qui doivent être suffisamment longues pour une mesure précise mais pas trop pour éviter le regroupement de voisins trop éloignés n'appartenant pas à la même ligne de texte. La transformée de Hough est aussi une méthode très utilisée (voir [Pal1996]), chaque petit segment de l'image permet d'estimer les coefficients du vecteur directeur de la droite qui le soutient. La direction de l'inclinaison du document correspond aux coefficients les plus représentés. Les histogrammes permettent également d'estimer cette inclinaison (voir [Bloomberg1995]) comme de segmenter l'image redressée en lignes (voir [Gatos1997], [Pal2001]). C'est cette approche qui est présentée ici.

Définition 3.3.1 : histogramme
L'histogramme d'une image selon une direction α est une projection de cette image sur une droite parallèlement à une droite de vecteur directeur $d = \binom{1}{tan\alpha}$. Concrètement, si I est une image de dimension (X, Y), un histogramme est un vecteur dont chaque élément contient le nombre de pixels noirs sur une ligne de direction d tracée avec un algorithme comme celui de [Bresenham1965] (voir également [Bresenham1977]).

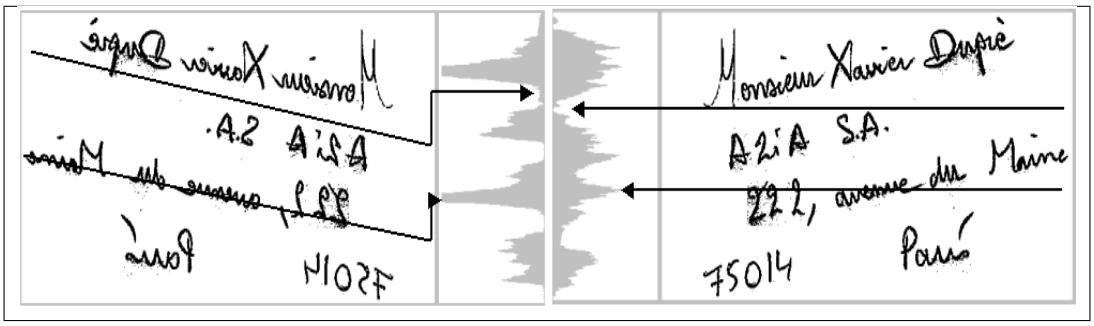


Fig. 3.7: Segmentation en lignes : recherche de la meilleure orientation, celle-ci accentue le plus possible les extrema.

La qualité de l'histogramme ou sa pertinence est estimée par son entropie.

Définition 3.3.2 : entropie d'un histogramme
Soit $H = (h_1, \ldots, h_n)$, on définit le vecteur défini par $H' = (p_1, \ldots, p_n)$:

$$\forall i \in \{1, \cdots, n\}\,,\; p_i = \frac{h_i}{\sum_{k=1}^{n} h_k}$$

L'entropie de l'histogramme H est le nombre suivant calculé sur l'histogramme H' :

$$E(H) = E\left(H'\right) = \sum_{i=1}^{n} p_i \ln p_i \tag{3.2}$$

La direction la plus probable est celle qui maximise l'entropie (voir [Côté1997]). Graphiquement, l'histogramme d'entropie maximale est celui dont les extrema sont les plus marqués (voir figure 3.7). L'image est finalement redressée de façon à ce que l'image ne contienne plus des lignes horizontales. Ce redressement peut tout simplement être effectué par un glissement des colonnes de pixels de l'image les unes par rapport aux autres.

Il existe des méthodes plus récentes comme par exemple celle décrite dans [Kapoor2004]. A partir d'une transformée de Radon de l'image et d'une transformée de Hough. Cette méthode est plus souple que la précédente. La méthode des histogrammes détermine l'orientation la plus probable dans un ensemble discret de solutions possibles. L'article [Kapoor2004] détermine directement cette meilleure orientation.

3.3.2 Segmentation en lignes

L'histogramme obtenu figure 3.7 est bruité. Afin de diminuer l'importance de ce bruit, l'histogramme est lissé par la méthode des moyennes mobiles. Selon les problèmes, la taille de cette moyenne est plus ou moins grande. Soit $H_l = (l_1, \ldots, l_n)$ l'histogramme lissé, il est donc obtenu à partir de H comme suit :

$$\begin{aligned} \forall i \in \{w+1, \cdots, n-w-1\}, \quad l_i &= \frac{1}{2w+1} \sum_{k=-w}^{+w} h_{i+k} \\ \forall i \in \{1, \cdots, w\}, \quad l_i &= \frac{1}{i+w} \sum_{k=1}^{i+w} h_k \\ \forall i \in \{n-w, \cdots, n\}, \quad l_i &= \frac{1}{n-i+w+1} \sum_{k=i-w}^{n} h_k \end{aligned} \tag{3.3}$$

Les maxima locaux indiquent la position des lignes, les minima locaux la position des frontières entre lignes. On définit pour chaque ligne les minima $(m_i^x)_i$ et les maxima $(M_i^x)_i$:

$$\forall i,\ m_i^x = \begin{cases} 1 \text{ si } l_i = \min\{l_k \mid l-x \leqslant k \leqslant l+x\} \\ 0 \text{ sinon} \end{cases}$$
$$\forall i,\ M_i^x = \begin{cases} 1 \text{ si } l_i = \max\{l_k \mid l-x \leqslant k \leqslant l+x\} \\ 0 \text{ sinon} \end{cases}$$

La figure 3.8 montre que bien souvent le nombre d'extrema détectés est supérieur au nombre réel d'extrema. Une étude sur quelques dizaines d'images a permis d'éliminer les cas de mauvaises détections les plus courants :

1. *Le petit palier* : ce cas se présente le plus souvent lorsqu'une ou plusieurs majuscules font partie de la ligne de texte. Le dessin de ces lettres contient des traits horizontaux tracés au-dessus de la ligne des minuscules. Une barre de "F" bien marquée peut entraîner de mauvaises segmentations.
2. *Le petit extremum* : lorsque les mots ne sont pas tout-à-fait bien alignés sur une même horizontale, les extrema sont plus diffus, il faut alors regrouper plusieurs maxima ensemble.

Deux règles permettent l'élimination de ces mauvaises détections :

1. Soit $\{e_i \mid 1 \leqslant i \leqslant 4\}$ quatre extrema consécutifs, alors :

$$|e_2 - e_3| \leqslant \beta\,|e_1 - e_4| \Longrightarrow \{e_2, e_3\} \text{ doivent être éliminés.} \tag{3.4}$$

2. Soit e_2 un minimum et e_1 et e_3 les extrema qui l'entourent, alors :

$$e_2 \leqslant \gamma \min\{e_1, e_3\} \Longrightarrow \{e_2, e_1 \text{ ou } e_3\} \text{ doivent être éliminés.} \tag{3.5}$$

Ce processus est illustré par la figure 3.8. Les valeurs intéressantes pour les quatre paramètres w, x, β, γ sont :

$$\begin{aligned} w &= 4 \text{ pixels} \qquad \beta = 0,2 \\ x &= 4 \text{ pixels} \qquad \gamma = 0,5 \end{aligned}$$

Le processus de suppression des "faux" extrema nécessite plusieurs itérations, à chacune d'elle, le plus petit palier est isolé et supprimé, ensuite, l'opération est répétée pour les petits extrema. Le processus s'arrête lorsqu'il ne peut plus rien supprimer, autrement dit, lorsqu'aucun petit palier et aucun petit extremum n'a pu être trouvé.

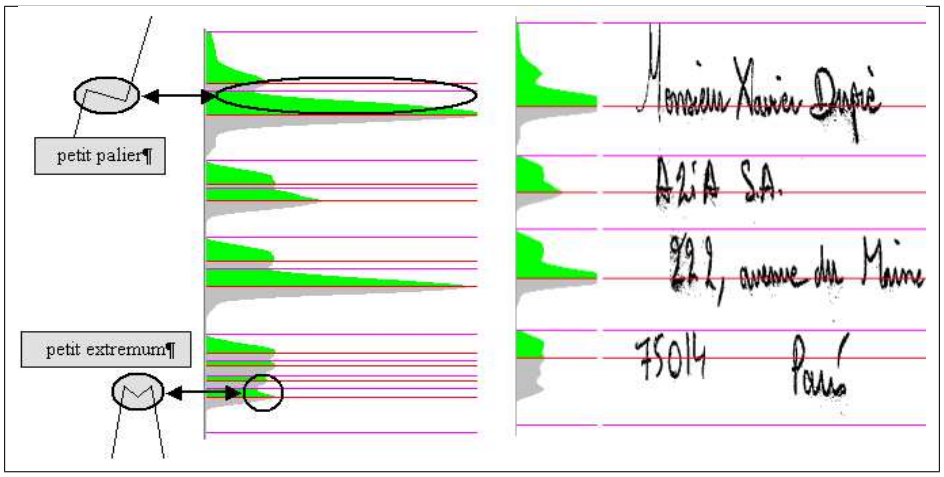


Fig. 3.8: Segmentation en lignes : recherche des bons extrema. Les extrema trop proches vérifiant les critères (3.4) et (3.5) ne sont pas pris en compte.

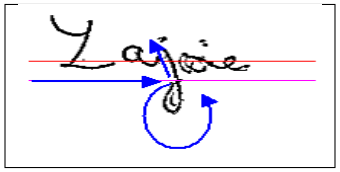


Fig. 3.9: Segmentation en lignes : connexité

3.3.3 Traitements des lignes enchevêtrées

C'est la dernière étape avant la reconnaissance du contenu des lignes. L'étude de l'histogramme a permis d'encadrer chaque ligne par un rectangle dont dépassent certaines grandes lettres ascendants et/ou descendants comme les "j" ou les "p". Le module de reconnaissance des mots est basé sur une extraction de graphèmes utilisant la connexité du dessin des lettres.

En partant de la même idée, on va supposer que le "j" de "Lajoie" (figure 3.9) est formé d'une seule composante connexe. La segmentation en lignes s'achève donc par le recollement des morceaux d'une même lettre égarés des deux côtés d'une frontière séparant deux lignes. Le principe est le suivant :

1. On parcourt la frontière entre deux lignes jusqu'à ce qu'on intercepte une lettre.
2. On parcourt le contour extérieur du morceau situé au-dessus, si lors de ce parcours, on revient à la même frontière sans en rencontrer aucune autre, alors ce morceau de lettre est considéré comme étant du mauvais côté.
3. On parcourt le contour extérieur du morceau situé au-dessous, si lors de ce parcours, on revient à la même frontière sans en rencontrer aucune autre, alors ce morceau de lettre est considéré comme étant du mauvais côté.
4. Si un seul des deux morceaux est du mauvais côté alors ce morceau est remis dans la bonne ligne, sinon on ne fait rien.
5. On continue le parcours de la frontière au cas où d'autres lettres intercepteraient celle-ci.

Dorénavant, l'extraction des lignes est terminée. Cette méthode fonctionne efficacement sur des adresses mais possède quelques écueils récurrents (figure 3.10).

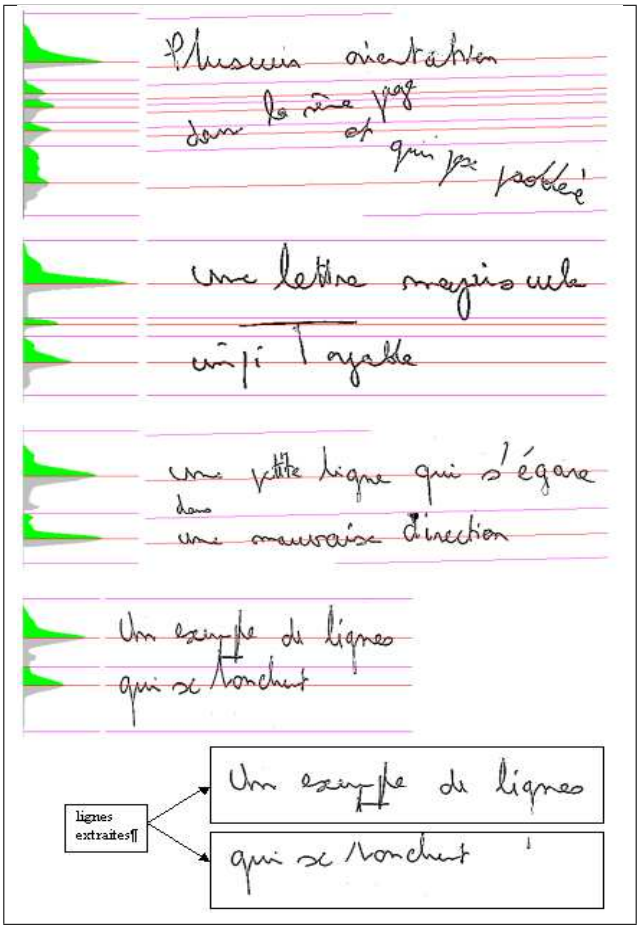


Fig. 3.10: Segmentation en lignes : exemples qui ne marchent pas.

3.3.4 Segmentation à partir d'un graphe

L'article [Abuhaiba1996] propose une autre alternative, une méthode de segmentation en lignes basée sur un graphe k-connexe (voir figure 3.11). L'image d'un paragraphe est d'abord squelettisée puis vectorisée [4]. Chaque arc ainsi obtenu est ensuite relié à k-plus proches voisins ordonnés selon la distance (3.6). Soient deux segments S_1 et S_2, la distance $d(S_1, S_2)$ est définie par :

$$\begin{aligned} d_x(S_1, S_2) &= \min_{(u,v)\in S_1\times S_2} |u_x - v_x| \\ d_y(S_1, S_2) &= \min_{(u,v)\in S_1\times S_2} |u_y - v_y| \\ d(S_1, S_2) &= \left[1 + \gamma \left(\frac{\pi}{2}\right)^{-1} \arctan \frac{d_y(S_1, S_2)}{d_x(S_1, S_2)}\right] \sqrt{d_x(S_1, S_2)^2 + d_y(S_1, S_2)^2} \end{aligned} \tag{3.6}$$

Le paramètre γ est choisi de telle sorte que deux segments appartenant à la même ligne soient plus proches que deux segments situés sur deux lignes consécutives. Chaque segment est donc relié à ses k plus proches voisins par un arc dont le poids est la distance (3.6). A ce graphe est ajouté un axe constitué de petits segments très peu éloignés de sorte qu'une connexion à cet axe est beaucoup moins coûteuse que tout autre connexion. L'arbre est ensuite réduit à un arbre de poids minimal en appliquant l'algorithme de

4. Annexes : voir paragraphe B.7.2, page 181

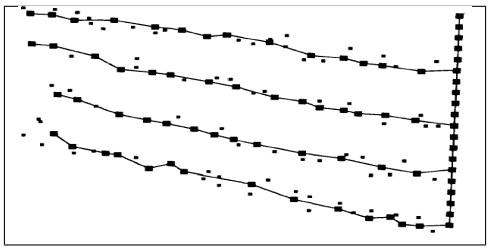


Fig. 3.11: Segmentation en lignes en recherchant l'arbre de poids minimal. L'axe vertical sur la droite de l'image est ajouté de façon à relier toutes les lignes entre elles, figure extraite de [Abuhaiba1996].

Kruskal (voir [Kruskal1956]). La segmentation en lignes s'achève par la détection de toutes les liaisons à l'axe virtuel non détruit par l'algortihme de Kruskal.

3.4 Prétraitements de l'image

Ce paragraphe regroupe ensemble différents prétraitements précédant une segmentation en graphèmes, il regroupe le redressement d'image ou l'estimation de différents paramètres comme la largeur moyenne d'une lettre, son épaisseur moyenne. Contrairement à la segmentation en lignes, ces méthodes sont particulières à l'écriture romaine. Les caractères chinois par exemple présentent des "imperfections" qui leur sont propres et qui nécessitent des traitements différents.

3.4.1 Redressement de l'image

Une fois les lignes d'un paragraphe extraites, il est parfois utile de redresser son image lorsque le scripteur a écrit "penché". Le rapport [Slavik2000] compare les performances en reconnaissance sur des images redressées ou non et montre l'apport substantiel des méthodes de prétraitement d'image. Les méthodes de redressement diffèrent bien sûr par leurs méthodes d'estimation de l'inclinaison mais aussi par les régions de l'image utilisées pour effectuer cette estimation.

L'inclinaison d'un mot est surtout visible pour les lettres possédant des ascendants et des descendants et c'est a priori cette partie de l'image qui doit être utilisée pour l'estimation de l'inclinaison comme le suggère la méthode de [Bozinovic1989] (voir figure 3.12) qui sélectionne les ascendants et descendants situés dans les parties supérieure et inférieure de l'image.

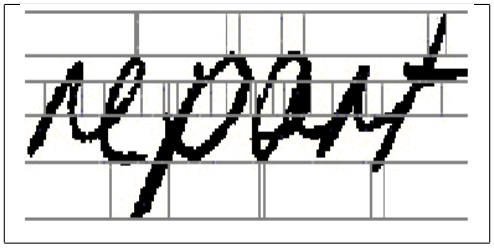


Fig. 3.12: Méthode de Bozinovic et Srihari (voir [Bozinovic1989]), figure exraite de [Vinciarelli2000]. Le principe de cette méthode consiste à estimer l'orientation du texte en ne considérant que les ascendants ou descendants suffisamment grands compris entre deux lignes verticales reliées aux bords supérieur et inférieur de l'image.

La méthode décrite dans [Slavik2000] quant à elle mesure l'inclinaison de chaque bord latéral des différentes composantes connexes puis en fait la moyenne pondérée par la longueur des segments obtenus. [Kim1997] et [Knerr1997] propose une estimation fondée sur les contours. Si n_+ et n_- désignent le nombre

de déplacements positifs et négatifs selon l'axe des abscisses, et n_v le nombre de déplacements verticaux, l'angle θ de l'inclinaison est obtenu à partir de l'expression de sa tangente : $\tan\theta = \frac{n_+ - n_-}{n_v + n_+ + n_-}$. La méthode de [Vinciarelli2000] utilise des histogrammes. Pour différentes valeurs d'angles α, on calcule le nombre $H_\alpha(x) = \frac{h_\alpha(x)}{\Delta y_\alpha(x)}$, où $h_\alpha(x)$ est le nombre de pixels sur la colonne x et $\Delta y_\alpha(x)$ la distance qui sépare les pixels noirs situés le plus haut et le plus bas. $H_\alpha(x)$ vaut 1 uniquement si la colonne est constituée d'un seul segment. Ensuite, pour chaque valeur d'angle, on calcule $S(\alpha) = \sum_{x, H_\alpha(x)=1} h_\alpha^2(x)$. L'inclinaison du mot est alors l'angle α qui maximise $S(\alpha)$. Toutes ces méthodes sont mieux adaptées à la détection de l'inclinaison de l'écriture romaine, d'autres types d'écriture, comme le montre l'article [You2003] dans le cas de l'écriture coréenne, nécessitent des développements plus spécifiques.

La méthode proposée ici s'inspire de celle décrite dans [Yanikoglu1998] et s'avère plus simple que l'estimation d'histogrammes. Elle donne les mêmes résultats qu'une méthode estimant l'inclinaison à partir du contour sans pour autant chercher à les obtenir. Les deux filtres de Sobel (3.7) (voir [Prewitt1970]) permettent de retrouver en chaque point de l'image la direction du gradient.

$$F_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad \text{et} \quad F_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \tag{3.7}$$

Soit I l'image d'un mot, On note $G_x = I * F_x$ et $G_y = I * F_y$ les produits de convolution de l'image par les deux filtres décrits en (3.7). Il est alors possible de déterminer en un point (x, y) la direction du gradient de la façon suivante en s'arrangeant pour que celle-ci appartienne à l'intervalle $[0, \pi[$:

$$\theta(x,y) = \arctan\left[\frac{G_y(x,y)}{G_x(x,y)}\right] \in [0, \pi[\tag{3.8}$$

Cet intervalle est ensuite divisé en n sous-intervalles de longueur identique afin de construire l'histogramme $(\alpha_i)_{1 \leqslant i \leqslant n}$ suivant :

$$\forall i \in \{1, \ldots, n\},\ \alpha_i = card\{(x,y) \in I \mid \theta(x,y) \in [0, \pi[\}$$

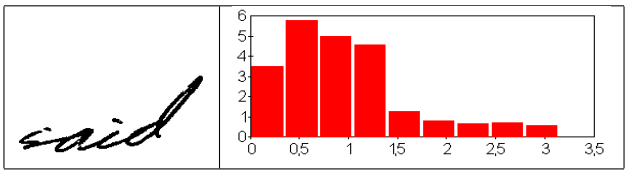


Fig. 3.13: Répartition de la direction du gradient pour une image de mot en neuf intervalles d'angle (les angles sont en radians). Le pic de l'histogramme est décalé par rapport à $\pi/2$.

Un exemple d'un tel histogramme est donné par la figure 3.13. Il montre un pic principal qui correspond à l'orientation des lettres hautes (t,l,...). Toutefois, à partir de quelques images, on a constaté que cet histogramme mène à une estimation moins précise de la direction d'inclinaison qu'une moyenne sur l'ensemble des directions calculées en évitant les directions proches de l'horizontale. On note $\hat{\theta}$ cette estimation :

$$\hat{\theta} = \frac{1}{\sum\limits_{(x,y)} \mathbb{1}_{\left\{\theta(x,y) \in \left[\frac{\pi}{8}, \frac{7\pi}{8}\right]\right\}}} \sum_{(x,y)} \theta(x,y)\, \mathbb{1}_{\left\{\theta(x,y) \in \left[\frac{\pi}{8}, \frac{7\pi}{8}\right]\right\}} \tag{3.9}$$

L'image est ensuite redressée en faisant glisser les lignes de pixels les unes sur les autres comme le montre la figure 3.14.

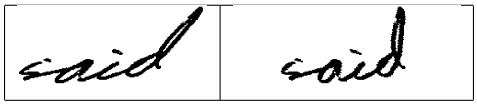


Fig. 3.14: Correction de l'inclinaison de l'image effectuée par un glissement des lignes les unes sur les autres. L'estimation de la direction par (3.9) donne 58 degrés, 90 étant la valeur pour écriture non inclinée.

3.4.2 Lissage du contour

La correction de l'inclinaison se termine par la construction de l'image corrigée qui est le résultat d'un glissement des lignes de pixels les unes par rapport aux autres. Cette méthode simple a pourtant l'inconvénient de produire des irrégularités tout le long du contour des lettres (voir figure 3.15). Ces petits bruits peuvent altérer les résultats de la reconnaissance (voir [Slavik2000]).

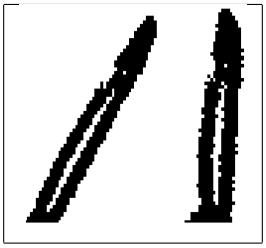


Fig. 3.15: Ces deux images proviennent de la barre du "d" de la figure 3.14. Le résultat obtenu dû aux glissements des lignes les unes par rapport aux autres présente de nombreuses irrégularités qu'il serait préférable de gommer.

La méthode proposée dans [Slavik2000] s'appuie sur la même méthode que celle qui permet d'obtenir le squelette d'une image[5]. Par l'application des masques de la figure 3.16, les lettres dont l'inclinaison a été corrigée perdent peu à peu leurs petites rides. Le processus continue tant que l'image évolue. Cet algorithme a été utilisé pour lisser l'image de la figure 3.17 dont l'inclinaison a été corrigée. La figure 3.17c montre le résultat obtenu grâce à cet algorithme de lissage utilisant les masques cités par la figure 3.16. Le résultat est satisfaisant, les lignes droites incluent néanmoins de larges créneaux qu'il serait possible d'élaguer en étudiant la convexité du contour, en minimisant le nombre de ses points d'inflexion.

3.4.3 Lignes d'appui

Les lignes d'appui encadrent la bande des minuscules et délimitent les zones contenant les ascendants et descendants (voir figure 3.18). Plusieurs méthodes permettent de détecter ces lignes virtuelles mais toutes nécessitent quelques lettres afin de retourner un résultat fiable. Il est par exemple impossible de distinguer un "o" minuscule d'un "O" majuscule si aucune autre lettre qui serait juxtaposée ne vient aider la lecture.

L'article [Wang1997] propose une méthode utilisant ces mêmes extrema locaux du contour extérieur de l'image d'un mot mais les lignes d'appui sont estimées globalement sur toute l'image à partir d'une transformée de Hough. Comme les images sont supposées contenir deux lignes d'appui parallèles et proches, les résultats sont affinés afin d'obtenir cette configuration.

5. Annexes : voir paragraphe B, page 159

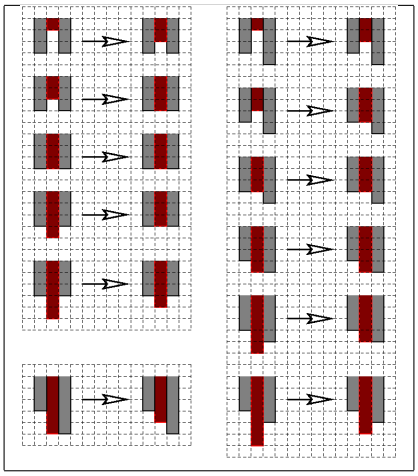


Fig. 3.16: Les configurations pixelliques ci-dessus permettent de lisser le contour après que l'inclinaison d'un mot a été corrigée. Cette figure est extraite de [Slavik2000] à laquelle il faut ajouter les configurations obtenues par rotation de celles présentées ci-dessus.

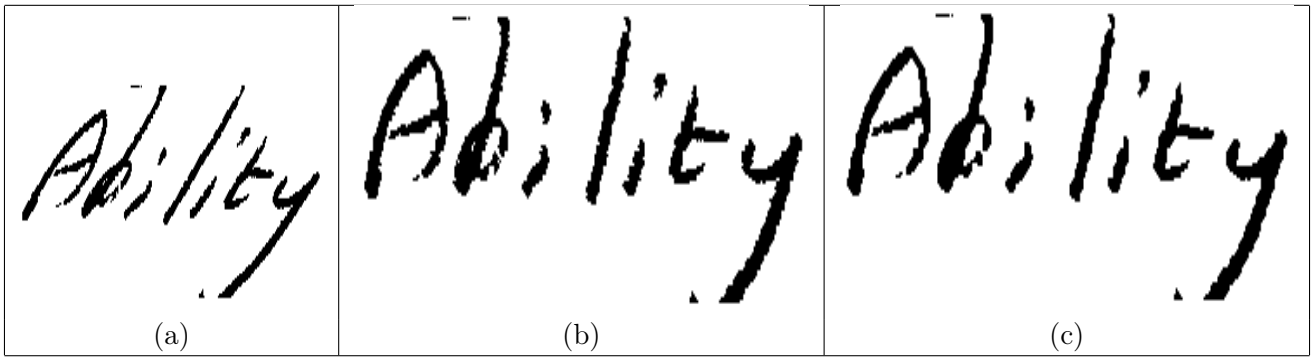


Fig. 3.17: L'inclinaison de l'image (a) est corrigée par la méthode exposée au paragraphe 3.4.1 et donne l'image (b). Les irrégularités du contour sont ensuite corrigées grâce à l'algorithme présenté au paragraphe 3.4.2 et qui aboutit à l'image (c).

Un autre article ([Madhvanath1999]) suggère que l'estimation globale de la position de ces lignes mène fréquemment à un résultat de mauvaise qualité surtout si les lettres ne sont pas disposées sur une droite ou si les minuscules présentent des tailles différentes. La méthode proposée dans cet article s'appuie sur les extrema du contour de l'image d'un mot puis regroupe localement ces points en petits segments regroupant des points proches et presque alignés. L'ensemble de ces petits segments définit des lignes d'appui variables tout au long du mot.

Plusieurs histogrammes peuvent être utilisés, épaisseurs des traits, nombre de transitions blanc-noir, projection des points du contour, celui-ci est souvent lissé par une moyenne mobile. En règle générale, la ligne d'appui basse est la plus fiable. Soit un histogramme $h = (h_1, \ldots, h_N)$, où h_1 correspond au bas de l'image et h_i est la moyenne des longueurs des segments blancs de la ligne i. L'histogramme est lissé avec une moyenne mobile analogue à (3.3). On définit l_b comme la ligne d'appui basse, l_h la ligne d'appui haute, l la ligne correspond au maximum de l'histogramme :

$$l \in \underset{i \in \{1,\ldots,N\}}{\arg\min}\ h_i \tag{3.10}$$

On définit ensuite l'intervalle $[l_b, l_h]$ autour de l vérifiant :

$$\forall i \in [l_b, l_h]\,,\ h_i \leqslant \alpha\, h_l \tag{3.11}$$

Le résultat de la figure 3.18 est obtenu pour $\alpha = 3$ ainsi que ceux de la figure 3.19. Ces formules peuvent être ajustées manuellement sur quelques images. Puisqu'elles sont basées sur des histogrammes, elles sont en général robustes. De plus, un écart de quelques pixels n'altère pas les résultats de la reconnaissance, l'essentiel est de définir un repère qui permette de positionner les lettres les unes par rapport aux autres à partir d'une origine définie par les lettres. La seconde ligne d'appui représente en quelque sorte un facteur d'échelle.

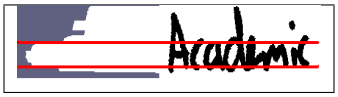


Fig. 3.18: Lignes d'appui encadrant la bande des minuscules

Fig. 3.19: Mauvais positionnement des lignes d'appui : les mots sont trop courts ou incluent des minuscules décorées comme la lettre "i". Ces cas sont minoritaires.

Afin de limiter les erreurs comme celles présentées figure 3.19, les algorithmes incluent parfois une pré-classification des histogrammes en quatre classes qui déterminent s'il faut chercher les lignes d'appui, une seule ou aucune : (voir [Hennig2002])

1. mot sans ascendant, sans descendant
2. mot avec ascendant(s), sans descendant
3. mot sans ascendant, avec descendant(s)
4. mot avec ascendant(s), avec descendant(s)

Le principe exposé ci-dessus est valable essentiellement pour des images de mots. Pour une ligne entière composée de plusieurs mots, même si la méthode ci-dessus peut servir de première approximation, elle doit être affinée pour chacun des mots en utilisant les boîtes englobantes des graphèmes par exemple (paragraphe 3.5). L'article [Hennig2002] présente une autre méthode basée sur des splines résolvant ce problème.

3.4.4 Estimation de l'épaisseur du tracé

Selon les scripteurs, l'écriture peut-être plus ou moins épaisse (voir figure 3.20). Cette différence n'est pas toujours intéressante à prendre en compte (redressement de l'image) comme elle peut parfois être une donnée non négligeable. Par exemple, on considère un histogramme de projection verticale utilisé pour

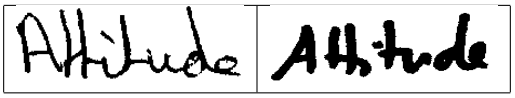


Fig. 3.20: Différentes épaisseurs de tracé pour deux images dont les dimensions sont 206x69 pixels pour la première et 211x97 pixels pour la seconde.

la segmentation graphème (paragraphe 3.5.5), ses minima locaux sont a priori supérieurs à l'épaisseur du trait qui peut servir de seuil de coupure.

L'article [Abuhaiba1996] propose une estimation à partir d'une carte de distance [6] et d'un masque de distance défini comme suit :

4	3	4
3	0	3
4	3	4

On note d_* la valeur de la distance la plus fréquente, l'épaisseur du trait $\widehat{e_0}$ est alors définie comme :

$$\widehat{e_0} = \frac{2}{3}\, d^* - 1 \tag{3.12}$$

Cette estimation peut être aussi effectuée au moyen d'histogrammes de projection. Une ligne ou une colonne de pixels extraite de l'image est constituée d'une suite de segments de la couleur de l'écriture. A priori, la longueur minimale de ces segments est égale à l'épaisseur du trait. Par conséquent, on construit l'histogramme $(h_i)_{1 \leqslant i}$ suivant :

$$\forall i \geqslant 1,\ h_i = card\,\{\text{ segments de longueur } i\ \} \tag{3.13}$$

La figure 3.21 illustre les histogrammes obtenus pour les deux images de la figure 3.20. La longueur (3.14) correspondant au maximum est une première estimation de l'épaisseur du trait. Un second estimateur (3.15) est construit à partir de celui-ci dans le cas où on considère que la distribution de la longueur des segments suit grossièrement une loi normale autour de cet extremum :

$$\widehat{e_1} = \arg\max_{i \geqslant 1} h_i \tag{3.14}$$

$$\widehat{e_2} = \frac{1}{h} \sum_{i=1}^{2\widehat{e_1}} i\, h_i \tag{3.15}$$

	première image	seconde image
$\widehat{e_1}$	4	10
$\widehat{\sigma\,(\widehat{e_2})}$	0,15	0,20

Tab. 3.1: Valeurs obtenues par les deux estimateurs définis en (3.14) et (3.15) pour les deux images de la figure 3.20.

Cette épaisseur est calculée pour les deux images de la figure 3.20 dans la table 3.1. Nécessairement, la largeur d'une lettre dépasse l'épaisseur du trait et on peut vraisemblablement penser que la largeur des lettres est au moins supérieure à deux fois cette épaisseur (voir [Yanikoglu1998]).

6. Annexes : voir paragraphe B.3, page 162

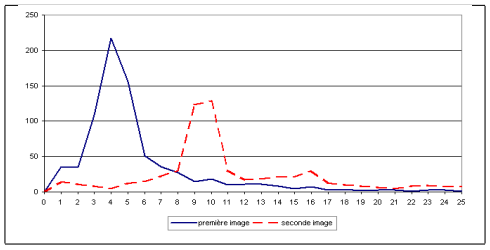


Fig. 3.21: Histogramme de répartition des longueurs des traits pour les deux images de la figure 3.20.

3.4.5 Estimation de la largeur moyenne d'une lettre

La largeur d'une lettre peut être une information intéressante à prendre en compte lors de la segmentation en graphèmes. Cette grandeur est d'abord estimée par la longueur moyenne entre deux transitions pixel noir - pixel blanc dans la bande des minuscules délimitées par les deux lignes d'appui estimées au paragraphe 3.4.3. Cette estimation est notée e_l par la suite.

3.4.6 Nettoyage de l'image

Les illustrations représentent souvent des images binaires où seul le mot à reconnaître apparaît. Toutefois, ces images "propres" sont rarement celles immédiatement obtenues après scannerisation du document. Il n'existe pas de méthodes générales associées à ces nettoyages car ils dépendent fortement du type de documents à traiter et des informations qui doivent y être reconnues. Les algorithmes développés sont donc spécifiques à un type précis de document (chèque, ordonnance, feuille de maladie, ...). Néanmoins, il se dégage trois catégories de prétraitements : la binarisation (voir [Kwon2004]) ou tout traitement d'image global, la localisation ou la recherche de l'information à reconnaître et à extraire, le nettoyage proprement dit qui consiste à enlever tout ce qui peut gêner la reconnaissance de la partie extraite, c'est un traitement local.

La figure 3.22 est un exemple emprunté à une image en niveaux de gris dont le fond est foncé. La première étape consiste généralement à binariser l'image. Ce premier traitement n'est pas incontournable mais il permet de réduire fortement la taille des images lors de la constitution de grandes bases de données et d'utiliser des algorithmes fondés sur la connexité. L'image de la figure 3.22 est déjà le résultat d'une extraction dont il faut ensuite enlever le trait de soulignement et les formes situées au-dessous du mot. Le résultat de ces prétraitements correspond à la seconde image de la figure 3.22.

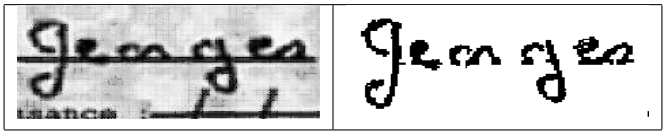


Fig. 3.22: Un mot extrait d'une page en niveaux de gris, avant de pouvoir reconnaître le mot, il faut binariser l'image et extraire le mot à reconnaître ce qui revient à enlever le soulignement et les divers lettres ou trait situés en-dessous. La seconde image est issue de la première après avoir été nettoyée.

Les procédures de nettoyage du trait de soulignement consiste d'abord à estimer son épaisseur puis à enlever les pixels qui le composent en prenant soin de laisser les pixels communs aux lettres et au trait de soulignement. Ces derniers sont fréquemment repérés par une zone de sur-épaisseur due au chevauchement des traits.

La détection des traits n'est pas non plus un problème simple même si, pour certains documents, le trait de soulignement est toujours présent (écriture d'un nombre de famille sur une ligne horizon-

tale par exemple). Il est possible d'utiliser des méthodes à base d'histogrammes comme ceux présentés au paragraphe 3.3. Une autre méthode intéressante est présentée dans les articles [Desolneux2000], [Desolneux2002], [Desolneux2003] qui propose un formalisme adapté à la détection de toute figure géométrique simple comme un segment, un carré, un cercle. Par exemple, un alignement de segments comme celui de la figure 3.23 n'est détecté que s'il est suffisamment isolé pour que sa présence ne puisse pas être considérée comme une coïncidence. En résumé, à partir des segments présents dans l'image, on quantifie d'abord la probabilité d'obtenir un alignement quelconque de petits segments n'importe où dans l'image ou plutôt le nombre moyen de segments faisant partie d'un alignement. S'il existe un ensemble de segments alignés supérieur au seuil déterminé juste avant, alors, on considère que cet alignement est plus que probable.

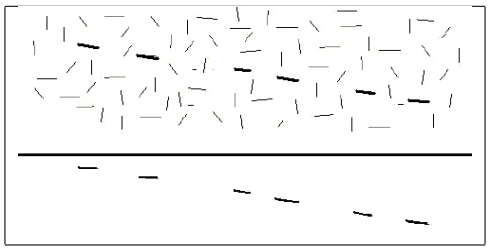


Fig. 3.23: Figure extraite de [Desolneux2002], les traits isolés présents au bas de la figure paraissent alignés mais noyés dans le nuage au-dessus, ils deviennent "invisibles".

Une méthode plus élaborée décrite dans [Cheng2004] permet de débarrasser l'image de mots manuscrits d'une ligne ou d'une courbe sur laquelle les lettres s'appuient, ou une courbe qui traverse l'image comme celle de l'exemple de la figure 3.24. Cette méthode s'appuie sur la construction d'un graphe qui résulte d'une squelettisation. La courbe principale découle d'une ou plusieurs recherche d'un plus court chemin.

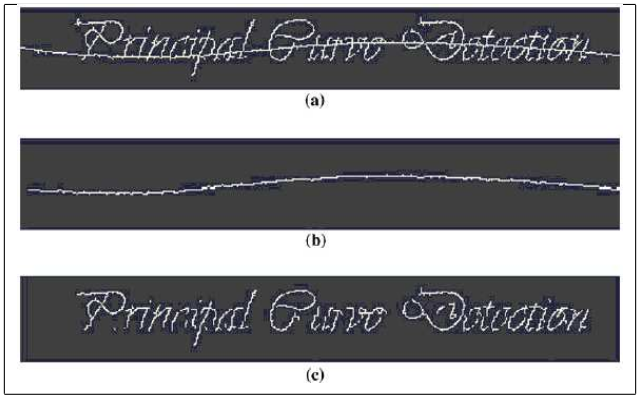


Fig. 3.24: Figure extraite de [Cheng2004], la première image (a) est l'image originale, les deux images suivantes résultent du nettoyage de cette première image, la courbe principale et les mots ont été séparées.

3.5 Diverses segmentations en graphèmes

La segmentation en graphèmes permet de délocaliser le problème de reconnaissance du niveau des mots au niveau des lettres. Reconnaître l'image d'un mot sans la découper au préalable est une méthode limitée à des problèmes restreints où le nombre de mots est faible comme l'écriture d'un nombre en lettres. Comme la reconnaissance se résume à un problème de classification. Plus la liste des mots possibles est longue, plus le classifieur à construire est complexe. C'est pourquoi il est préférable de scinder ce problème de

reconnaissance d'un mot en une somme de problèmes plus simples qui sont la reconaissance des lettres présentes dans l'image.

Découper l'image soulève plusieurs questions dont la première concerne le résultat à obtenir : est-il dépendant des modèles de reconnaissance utilisés par la suite ? Dans le cas de modèles de Markov cachés, le résultat souhaité est une séquence d'observations, ce qui signifie que la seule dimension variable du découpage est le nombre d'objets ainsi formés. Une extension de ces modèles statistiques permet d'étendre le concept de séquence à un graphe d'observations incluant plusieurs options de segmentations. Toutefois, quelle que soit l'option choisie, elle résulte des compromis suivants :

1. Plus la segmentation possède de degrés de liberté (plus elle propose d'alternatives), plus les modèles de reconnaissance seront complexes, plus les modèles de reconnaissance sont complexes, plus ils sont difficiles à apprendre.
2. Moins la segmentation possède de degrés de liberté, plus elle est susceptible de faire des erreurs.

Nous allons aborder une segmentation sous forme de séquences de graphèmes. Cette étape de segmentation est indispensable pour la construction d'un système de reconnaissance de l'écriture, diverses méthodes sont passées en revue dans les articles [Lecolinet1991] ou plus récemment [Lu1996]. Les paragraphes qui suivent reprennent quelques-unes des méthodes présentées dans ces articles puis concluent sur la conception de la segmentation qui a été élaborée dans le cadre de ces travaux de recherche. Cette segmentation propose également une solution au problème des accents dont la lettre d'attache est parfois située assez loin, ce qui n'a pas été pris en compte jusqu'à présent.

3.5.1 Segmentation à partir du squelette

Les graphèmes sont des images extraites de l'image à segmenter. Passer d'une seule image à une séquence de graphèmes soulève deux problèmes (voir [Baret1991]) qui sont la taille que doivent avoir les graphèmes et l'ordonnancement des morceaux segmentés. Ils ne doivent pas être trop petits afin d'être différents les uns des autres, différents d'un simple trait. Ils ne doivent pas être trop gros pour ne pas dépasser la taille d'une lettre. Chaque lettre représentera entre un et trois graphèmes. Ce choix facilite l'ordonnancement des graphèmes qui doivent respecter le sens gauche-droite de la lecture.

L'image peut être rendue à l'état de squelette[7]. Ce dernier est ensuite parcouru de manière à repérer certains motifs synonymes de césure entre lettres (figure 3.25). La détection de ces motifs introduit des calculs de courbure, d'angle qui sont comparés à des seuils ajustés de manière à obtenir le résultat désiré. Ces algorithmes sont mieux détaillés dans [Lecolinet1991].

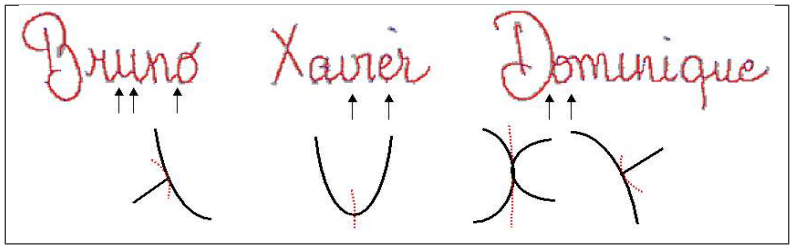


Fig. 3.25: Segmentation à partir du squelette : segmentation basée sur des motifs.

La figure 3.26 est un exemple de ce qui est obtenu et des problèmes qui accompagnent l'utilisation de seuils. Les lettres en fin de mots, plus petites, sont parfois agrégées. La figure 3.2 recense la liste de ces problèmes.

7. Annexes : voir paragraphe B, page 159

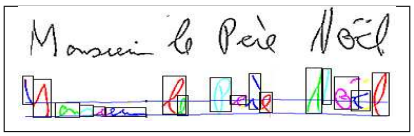


Fig. 3.26: Segmentation à partir du squelette : chaque graphème est entouré de sa boîte englobante, les deux lignes horizontales modélisent les lignes d'appui (ou lignes de bases) qui encadrent la bande où sont écrites les lettres minuscules (paragraphe 3.4.3).

3.5.2 Segmentation à partir du contour

Cette méthode esquissée dans [Madhvanath2001] ne s'intéresse pas au squelette mais uniquement au contour dont elle détermine les meilleurs points candidats à une coupure entre graphèmes (voir figure 3.27). Lors du parcours du contour, les extrema locaux sont marqués (point culminant et selle) puis les paires les plus proches sont regroupées de part et d'autre du trait.

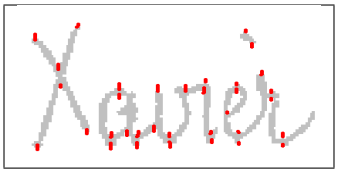


Fig. 3.27: Segmentation à partir du contour : les points représentent les minima et les maxima locaux (ordonnée des points) le long du contour. Les paires des points des plus proches disposés de part et d'autre du trait forment les candidats les plus probables pour une césure.

La direction de coupure n'est pas toujours horizontale comme le montre la figure 3.28. A l'instar de la méthode précédente, la segmentation en graphèmes à partir du contour nécessite de nombreux ajustements avant de trouver les critères de décision. Cette mise au point par tâtonnements est le point commun de nombreux traitements d'images liées à l'écriture manuscrite. Faciles à ajuster lorsque la qualité de l'écriture est bonne (figure 3.27), ces prétraitements peuvent avoir des comportements tout-à-fait erratiques lorsque l'écriture est de mauvaise qualité (voir figure 3.34).

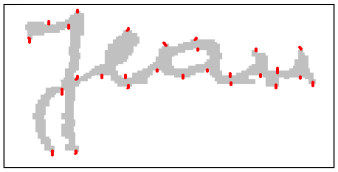


Fig. 3.28: Segmentation à partir du contour : si la césure du couple "Je" s'appuie sur les extrema environnants, sa direction est plus horizontale que verticale.

3.5.3 Ordonnancement

Une fois la segmentation effectuée, il ne reste plus qu'à ordonner les morceaux afin de former une séquence d'observations. Ce problème n'est pas simple et doit inclure des étapes de regroupement afin de traiter des problèmes tels que des accents qui doivent être associés à une lettre. Dans un premier temps, les accents, les points, c'est-à-dire tout morceau isolé au-dessus de la ligne d'appui haute, ne sont pas intégrés par l'algorithme d'ordonnancement, ils sont affectés aux graphèmes une fois ce dernier terminé. Ce problème d'ordonnancement est similaire au problème du voyageur de commerce. Une fois que les graphèmes de

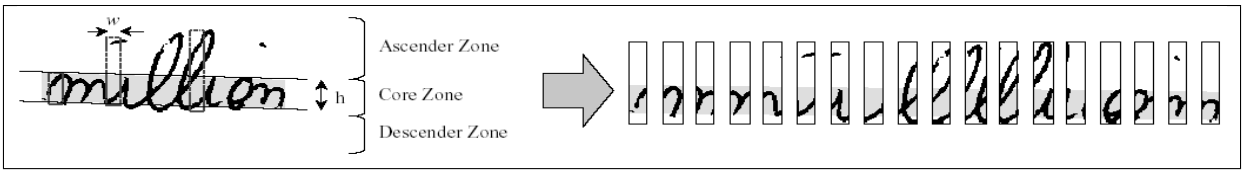


Fig. 3.29: Image extraite de [Knerr2001] illustrant le découpage de l'image d'un mot par des fenêtres glissantes.

début et de fin sont déterminés, le plus court chemin reliant les graphèmes de début et de fin et incluant tous les autres graphèmes peut être assimilé à la séquence la plus probable.

3.5.4 Fenêtres glissantes

Découper l'image en bandelettes verticales est la segmentation la plus simple comme le montre la figure 3.29. Ce découpage peut être régulier ou dépendre des minima d'un histogramme de projection par exemple, il peut également se recouvrir. L'inconvénient de cette méthode est qu'elle génère trop de graphèmes regroupant les morceaux de plusieurs lettres, c'est d'ailleurs pourquoi les petites images obtenues se recouvrent en partie. Le paragraphe suivant 3.5.5 étend cette méthode à plusieurs directions de projection pour les histogrammes. Cette représentation est utilisée par le système de reconnaissance décrit dans [Knerr2001].

Cette méthode comme la suivante d'ailleurs possède néanmoins l'avantage par rapport à celle présentée dans les paragraphes 3.5.1 et 3.5.2 de n'être pas dépendante de la connexité et d'être moins sensible au bruit. L'ordonnancement (voir paragraphe 3.5.3) est lui aussi évident puisque la segmentation basée sur le squelette ou le contour produit des morceaux répartis dans un espace en deux dimensions alors que cette méthode segmente l'image selon l'axe des abscisses qui est aussi l'axe de lecture.

3.5.5 Segmentation basée sur des histogrammes

Cette méthode est décrite dans l'article [Yanikoglu1998] et produit une segmentation semblable à celle illustrée figure 3.30. Elle consiste à déterminer des droites de segmentation de l'image à partir d'histogrammes de projection effectués selon différentes directions proches de la verticale. Ces droites sont choisies de telle sorte qu'elles interceptent le moins de pixels noirs et sont régulièrement espacées dans l'image. Cette méthode simple et peu dépendante de la connexité ne peut malgré tout pas tout résoudre comme en témoignent les exemples de la figure 3.31. La méthode possède également l'avantage de ne pas être assujettie au problème d'ordonnancement (voir paragraphe 3.5.3).

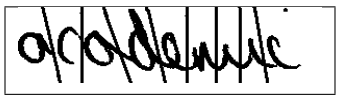


Fig. 3.30: Segmentation en graphèmes à partir d'histogrammes de projection selon plusieurs directions.

3.5.6 Segmentation basée sur des réservoirs

Cette idée est dévelopée dans l'article [Pal2003] et est appliquée dans le cadre d'une segmentation de chiffres cursifs. Elle consiste à détecter tout d'abord les vallées et les collines séparant deux chiffres appartenant à la même composante connexe, ces formes sont illustrées pour un mot dans la figure 3.32. Deux chiffres liés

Fig. 3.31: Deux exemples où la segmentation par histogramme est difficilement applicable : le premier cas contient une barre de "t" qui sera nécessairement coupée puisqu'elle touche la lettre "a". Le second exemple contient le couple "op" fortement lié du fait de l'épaisseur du trait, la liaison entre les deux lettres est plus épaisse qu'ailleurs. Le dernier mot présente un couple "La" qu'aucune droite ne saurait séparer.

dans une même composante seront séparés si une vallée ou une colline représente un espace suffisamment grand par rapport à la taille des chiffres. En ce qui concerne les lettres, les règles de décision sont plus difficiles à mettre en place car les lettres ont des hauteurs variables.

Fig. 3.32: Segmentation à partir de réservoirs d'eau ([Pal2003]) : les vallées et les collines sont en quelque sorte remplies d'eau, si elles sont suffisamment profondes ou hautes, elles marquent la séparation entre deux caractères.

Une fois que les zones de coupures sont détectées, il reste à déterminer à quelle catégorie elle appartient (voir figure 3.33) afin de placer la césure à l'endroit le plus approprié. Cette idée est reprise dans [Elnagar2003] à ceci près que la méthode s'applique au squelette des chiffres et pas à l'image initiale.

Fig. 3.33: Segmentation à partir de réservoirs d'eau ([Pal2003]) : deux points de coupures différents, le premier est situé au fond d'une vallée à droite sur un embranchement, le second est situé dans un creux, au milieu d'une courbe en "u". La coupure ne doit donc pas toujours intervenir à l'endroit du minimum local.

3.5.7 Graphes de graphèmes

La segmentation en graphèmes donne parfois des résultats erronés. La figure 3.34 propose une manière d'éviter ces erreurs en résumant au travers d'un graphe plusieurs options de segmentation. La séquence de graphèmes est un cas particulier de ce graphe, elle est la segmentation la plus probable pour la partie qui concerne le traitement d'image mais pas forcément la meilleure pour la partie reconnaissance qui suit. Il peut donc être intéressant de proposer plusieurs segmentations afin d'augmenter la probabilité que la bonne segmentation soit trouvée. Ce procédé revient souvent comme un leitmotiv dans la reconnaissance de l'écriture, il s'agit de retarder la prise de décision afin de conserver à chaque étape le plus de solutions possibles.

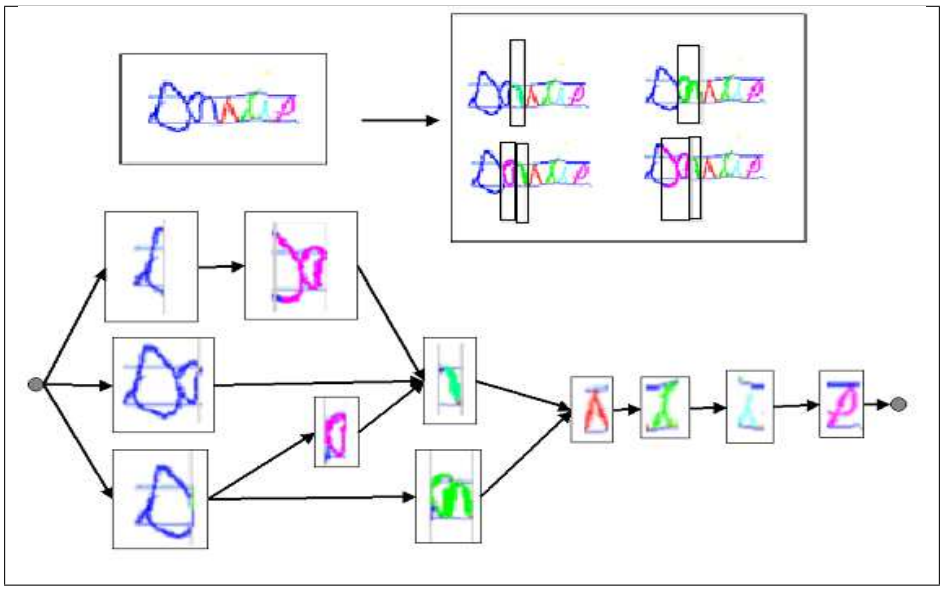


Fig. 3.34: Segmentation en graphèmes : graphes. Aucune décision n'est prise à ce niveau, le choix de la bonne segmentation sera effectué par le module de reconnaissance.

3.6 Choix d'une segmentation en graphèmes

La segmentation décrite dans les paragraphes qui suivent fonctionne bien lorsque l'écriture est de bonne qualité. Comme tous les algorithmes de segmentation fondés sur des heuristiques, celui-ci ne peut traiter correctement la totalité des images. Cependant sa construction met en lumière les difficultés rencontrées lorsque la qualité de l'écriture décroît et les ajustements rendus nécessaires par des problèmes récurrents.

3.6.1 Segmentation à partir d'histogrammes

Le choix d'une segmentation dépend des modèles de reconnaissance qui devront l'utiliser. Segmenter en lettres ou morceaux de lettres les mots illustrés par la figure 3.31 ou 3.35 peut paraître une gageure. Toutefois le paragraphe 4.7 permet d'assouplir cette contrainte. La segmentation proposée ici vise seulement le découpage d'un mot en morceaux pouvant aller d'une simple partie de lettre à des groupes de deux ou trois lettres, pourvu que ceux-ci soient aisément identifiables. L'objectif est aussi d'éviter le plus possible les traitements basés sur la connexité car ils sont très sensibles au bruit.

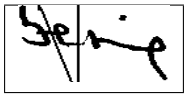


Fig. 3.35: Segmentation en graphème d'un mot couramment employé en langue anglaise : *being*. Les trois dernières lettres "ing" sont simplement esquissées.

L'impossibilité de qualifier la pertinence d'une segmentation demeure un inconvénient majeur, il est en effet difficile d'apprécier directement un prétraitement de l'image dont on attend les bénéfices seulement en fin de chaîne, c'est-à-dire en terme de taux de reconnaissance. L'appréciation n'est donc que visuelle.

L'approche proposée ici est un compromis. La première étape consiste à segmenter grâce à la méthode des histogrammes (paragraphe 3.5.5) en ne conservant que des coupures évidentes. Un premier ensemble de points de coupures est ainsi obtenu parmi lesquels seront sélectionnés ceux définissant la segmentation

en graphèmes finale. Ce dernier résultat n'est pourtant pas encore parfait, ce que tenteront de corriger les méthodes basées sur les contours ou les réservoirs, celles-ci permettront ensuite de couper les morceaux litigieux.

Tout d'abord, les pixels sont projetés selon sept directions entourant la direction verticale -30^o, -20^o, -10^o, 0^o, 10^o, 20^o, 30^o. On note $(h_{ij})_{\substack{-3\leqslant i\leqslant 3\\ 1\leqslant j\leqslant X}}$ les sept histogrammes obtenus où X est la longueur de l'image. h_{ij} est donc le nombre de pixels noirs (écrits) selon une droite formant un angle $i \times 10^o$ avec la verticale et interceptant la ligne d'appui basse au point d'abscisse j. Ces histogrammes sont ensuite lissés par une moyenne mobile analogue aux formules (3.3). On définit e_t comme étant l'épaisseur du tracé (paragraphe 3.4.4), e_l correspond à la largeur moyenne d'une lettre estimée au paragraphe 3.4.5. Enfin C est l'ensemble de coupures et défini par :

$$C = \{h_{ij} \mid h_{ij} \leqslant \beta\, e_t\} \text{ où } \beta \geqslant 1 \tag{3.16}$$

Le paramètre β est généralement compris entre 1 et 2 de manière à ne pas couper un mot selon une droite interceptant deux fois le tracé. Il est possible d'écrire l'ensemble C comme une réunion d'intervalles.

$$C = \bigcup_{k=-3}^{3} \bigcup_{i=1}^{I} \left[a_i^k, b_i^k\right] \text{ avec } \left\{ \begin{array}{l} a_i^k \leqslant b_i^k < a_{i+1}^k \ \forall i,k \\ h_x^k \leqslant \beta\, e_t \ \forall x \in \left[a_i^k, b_i^k\right] \end{array} \right. \tag{3.17}$$

Pour chaque intervalle de la forme $\left[a_i^k, b_i^k\right]$, on vérifie que $b_i^k - a_i^k \leqslant e_l$. Dans le cas contraire, on scinde cet intervalle jusqu'à ce que chacun des morceaux soit inférieur à e_l. La figure 3.36 soulève le problème de soulignement. Etant donné la condition exprimée en (3.17), il est impossible de sélectionner une seule zone de coupure probable entre graphèmes. Par conséquent, la solution adoptée est l'introduction de points de coupure entre les zones de non-coupure correspondant à des minima locaux.

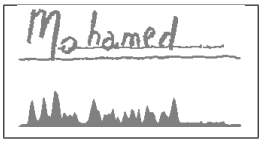


Fig. 3.36: Sélection des zones de coupures entre graphèmes : problème des mots soulignés, l'histogramme représenté correspond à une projection verticale lissée par une moyenne mobile uniforme d'ordre trois.

Dans ce cas, pour une direction donnée k, l'ensemble des points de coupures correspond aux minima locaux de l'histogramme $\left(h_i^k\right)_i$. Un minimum m^k local vérifie la condition suivante :

$$h_{m^k}^k = \min\left\{h_x^k \mid m^k - e_t \leqslant x \leqslant m^k + e_t\right\} \tag{3.18}$$

Ces minima locaux n'existent pas toujours, dans ces cas, on cherche à déterminer le point $c_i^k \in \left[a_i^k, b_i^k\right]$, l'unique point de coupure de la zone de coupure $\left[a_i^k, b_i^k\right]$. La figure 3.33 suggère que ce point se situe à droite du milieu de cet intervalle, par conséquent, $\tau_2 > \tau_4 > \tau_3$ dans la définition suivante :

$$\begin{aligned} m_i^k &= \frac{a_i^k + b_i^k}{2} \\ c_i^k &= \underset{x\in\left(a_i^k, b_i^k\right)}{\arg\min} \left[\tau_1 \frac{h_x^k}{e_t} + \frac{4}{e_l^2}\left[\mathbb{1}_{\{x<m_i^k\}}(\tau_2 - \tau_3) + \tau_3\right]\right] \left[x - m_i^k\right]^2 + \frac{2\tau_4\, e_t^2}{e_t^2 + \left|s_{x^-}^k - s_{x^+}^k\right|} \end{aligned} \tag{3.19}$$

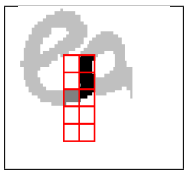


Fig. 3.37: Définition des nombres $s^k_{x^-}$ et $s^k_{x^+}$ de la formule (3.19). $s^k_{x^-}$ correspond au nombre de pixels contenus dans la première colonne du rectangle quadrillé (couleur gris foncé), $s^k_{x^+}$ correspond à la colonne de droite (couleur noire). Le côté des petits carrés est égal à e_t soit l'épaisseur moyenne de l'écriture.

Les nombres $s^k_{x^-}$ et $s^k_{x^+}$ sont définis par la figure 3.37. L'ensemble $\left\{c^k_i\right\}_{ik}$ est l'ensemble des droites de segmentation possibles sélectionnées par l'algorithme, cet ensemble est trié par i et k croissant (i d'abord, l'indice k départageant les points de même indice i) et aboutit à la suite $(d_n)_{1\leqslant n\leqslant N}$. Il reste à déterminer quelles sont parmi les points de cette suite les droites de segmentation les plus pertinentes.

A cette suite, sont ajoutés les éléments sélectionnés par l'équation (3.18) et deux éléments d_0 et d_{N+1} qui correspondent aux deux séparations verticales de début et de fin, c'est-à-dire les limites de l'image. On suppose qu'il existe une distance entre deux droites de coupures i et j notée $(D_{ij})_{0\leqslant i,j\leqslant N+1}$, trouver la meilleure segmentation revient alors à trouver le plus court chemin entre les nœuds d_0 et d_{N+1} en passant ou non par n autres nœuds $(d_n)_{1\leqslant n\leqslant N}$. Ce problème est usuel et résolu par un algorithme du plus court chemin de type Djikstra (voir [Dijkstra1971]). Il reste à déterminer la distance D_{ij} entre deux droites de coupures qui doit prendre en compte trois éléments :

1. Le nombre de pixels noirs interceptés par les droites de coupures, noté p_i et p_j.
2. Le fait que les droites s'interceptent ou non, noté $t_{ij} \in \{0, 1\}$ (voir figure 3.38).
3. La distance entre les deux points d'intersection des deux droites avec la ligne d'appui basse, notée $d_j - d_i$ cette distance devrait être proche de λ_4 fois la largeur supposée d'une lettre, notée e_l et calculée au paragraphe 3.4.5.

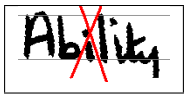


Fig. 3.38: Droites de coupures sécantes issues d'une segmentation non pertinente, deux droites de coupures peuvent se croiser, auquel cas, il n'est pas très pertinent de les associer ensemble pour former la segmentation en graphèmes.

A chaque abcisse d_i est associé un angle u_i qui correspond à la direction de l'histogramme qui a permis d'obtenir d_i. A l'aide de ces notations, la distance D_{ij} est définie par :

$$D_{ij} = \lambda_1 \frac{p_i + p_j}{e_t} + \lambda_2\, t_{ij} + \lambda_3 \frac{(d_j - d_i - \lambda_4 e_l)^2}{e_l^2} + \lambda_5\, (u_j - u_i)^2 \tag{3.20}$$

En pratique, λ_4 est choisi proche de 1, λ_2 est grand, λ_3 et λ_1 sont choisis proches de 1. Divers problèmes subsistent après ce traitement illustré par la figure 3.39, il reste des bouts de lettres mal appariés, des accents mal placés, des couples de lettres inséparables par une droite et pourtant formés de deux composantes connexes ou presque disjointes, des petits morceaux qu'on pourrait associer à un graphème voisin plus gros. Ce découpage plus fin s'effectue en deux étapes :

1. Découpage d'un graphème contenant deux formes reliées par un pont de pixels comme le couple "lf" de l'image *(c)* de la figure 3.39.

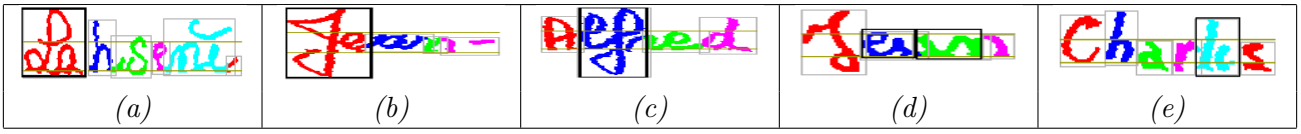


Fig. 3.39: Résultat intermédiaire de la segmentation graphème après obtention du meilleur chemin dans le graphe défini par la matrice d'adjacence (3.20). Les valeurs des paramètres utilisées pour cet exemple sont données par le tableau 3.2, page 68.

2. Découpage d'un graphème contenant plusieurs composantes connexes de tailles suffisantes pour être scindées en plusieurs graphèmes, image *(e)* de la figure 3.39.

3.6.2 Segmentation à partir de "réservoirs"

Comme le montre la figure 3.39, le traitement précédent laisse quelques imperfections qu'il serait possible de résorber en utilisant la méthode des réservoirs illustrée par la figure 3.32 et développée dans [Pal2003]. On considère les vallées et les collines dont la profondeur est supérieure à $\eta_1 e_t$ et la surface est supérieure à $\eta_2\, e_t\, (e_l - e_t)$. Il s'agit ensuite d'isoler les parties du tracé qui constituent le fond des vallées et des collines et susceptibles d'être coupées. Ce tracé correspond à la frontière d'une vallée ou d'une colline dont la largeur décroît, cette frontière inclut le fond de la vallée ou le sommet d'une colline noté (x^v, y^v). Si $e_t\,(x)$ est l'épaisseur du tracé à l'abscisse x, il est possible de choisir le point de coupure c_v en s'inspirant de l'équation (3.19) :

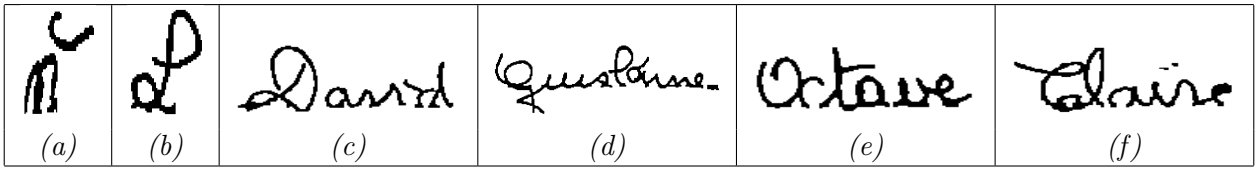
$$c^v = \underset{(x,y)\in vallée}{\arg\min} \left[\tau_1 \frac{e_t\,(x)}{e_t} + \frac{4}{e_l^2} \left[\mathbb{1}_{\{x<x^v\}} (\tau_2 - \tau_3) + \tau_3 \right] \left[x - x_v\right]^2 + \frac{\tau_5}{e_t} \left| y - y^v \right| \right] \tag{3.21}$$

L'algorithme décrit dans [Pal2003] s'applique à l'ensemble du mot, il coupe en deux l'image du mot, puis réitère le procédé pour chaque morceau obtenu jusqu'à qu'il ne puisse plus couper. Ce processus sera également appliqué à chaque graphème. Il reste à traiter le cas des vallées sans fond comme celle de la figure 3.40. Comme précédemment, si la largeur de cette vallée correspondant à la partie grisée est supérieure à $\eta_2 e_t\ (e_l - e_t)$, alors le graphème sera scindé.

Fig. 3.40: Segmentation à partir de réservoirs : vallée sans fond, la partie grisée correspond à la largeur de la vallée, si celle-ci est supérieure à $e_l e_t$, alors le graphème sera scindé en deux morceaux.

Vient ensuite le problème des accents qui se présente sous deux formes *image (a)* et *images (c)-(d)* de la figure 3.41. Dans le premier cas, il suffit de séparer deux composantes connexes en utilisant une vallée sans fond. Le second cas paraît impossible, le point de la lettre "i" vient toucher la lettre "d" (image *(c)*) ou la lettre "a" (image *(d)*). Le problème posé par l'image *(e)* ou *(f)* apparaît fréquemment, il s'agit de lettres dont les tracés parallèles se chevauchent, comme les couples "oc" ou "cl" des images *(d)* et *(e)* de la figure 3.41.

Parmi ces différents problèmes, seul le cas des accents appartenant à des composantes connexes différentes sera traité. Les autres n'apparaissent que pour des couples ou des groupes de lettres précis et seront



(a) *(b)* *(c)* *(d)* *(e)* *(f)*

Fig. 3.41: Segmentation à partir de réservoirs : cas des accents, il serait préférable que ceux-ci soient dissociés de la séquence de graphèmes car leur position est variable. Toutefois, aucune vallée ne permet de séparer l'accent (image *(a)*) à moins de considérer la transposée de l'image, mais dans ce cas, certaines lettres comme celle de l'image *(b)* seraient coupées par la méthode des réservoirs. De plus, comment détecter l'accent de l'image *(c)* ou celui de l'image *(d)* ? Comment traiter le problème des traits parallèles se chevauchant (images *(e)* ou *(f)* et des couples "oc" et "cl") ?

modélisés ultérieurement notamment (voir paragraphe 4.7, page 124). Il n'est pas non plus nécessaire de traiter des problèmes qui ne surviennent que peu fréquemment, des développements spécifiques risquent d'introduire de mauvais cas parmi ceux déjà bien traités. Un motif trop peu fréquent ne peut être appris par des modèles probabilistes tels que les chaînes de Markov cachées et les réseaux de neurones, et ce, qu'il soit bien ou mal segmenté.

L'algorithme qui suit permet de déterminer la profondeur des vallées, celle des collines s'en déduit facilement. On cherche la vallée la plus profonde et pour ce faire, on construit la matrice $v\left(x,y\right)_{\substack{1\leqslant x\leqslant X\\1\leqslant y\leqslant Y}}$ où (x,y)

est un pixel de l'image. Cette matrice est définie par l'algorithme suivant 3.6.1.

Algorithme 3.6.1 : profondeur des vallées, calcul de $v(x,y)$
On considère une image $I(x,y)$ de dimension (X,Y), on note la propriété qu'un pixel soit noir par $N(x,y)$. Le premier pixel est le coin supérieur gauche. $v(x,y)$ désigne la profondeur de la vallée.

Etape A : initialisation
pour $x = 1$ **à** X **faire**
$v(x,Y) \longleftarrow 0$
fin pour

Etape B : mise à jour
pour $y = Y - 1$ **à** 1 **faire**
pour $x = 1$ **à** X **faire**

$$v(x,y) \longleftarrow \begin{cases} 0 & \text{si } N(x,y) \\ v(x,y+1)+1 & \text{sinon} \end{cases}$$

fin pour
pour $x = 2$ **à** X **faire**

$$v(x,y) \longleftarrow \begin{cases} 0 & \text{si } \forall i \leqslant x,\ N(i,y) \text{ est faux} \\ max\{\, v(x,y),\ v(x,y+1)+1 \,\} & \text{sinon} \end{cases}$$

fin pour
pour $x = X - 1$ **à** 1 **faire**

$$v(x,y) \longleftarrow \begin{cases} 0 & \text{si } \forall i \geqslant x,\ N(i,y) \text{ est faux} \\ max\{\, v(x,y),\ v(x,y+1)+1 \,\} & \text{sinon} \end{cases}$$

fin pour **pour** $x = 2$ **à** X **faire**

$$v(x,y) \longleftarrow \begin{cases} 0 & \text{si } N(x,y) \\ max\{\, v(x-1,y),\ v(x,y) \,\} & \text{sinon} \end{cases}$$

fin pour
pour $x = X - 1$ **à** 1 **faire**

$$v(x,y) \longleftarrow \begin{cases} 0 & \text{si } N(x,y) \\ max\{\, v(x,y),\ v(x+1,y) \,\} & \text{sinon} \end{cases}$$

fin pour

fin pour

Le maximum atteint sur la première ligne de l'image correspond à la vallée la plus profonde.

En conservant en chaque point de l'image le pixel qui a permis d'atteindre le maximum, il est possible d'en déduire la surface de la vallée la plus profonde. L'algorithme peut être adapté de manière à estimer la surface de la colline la plus profonde. Il reste à déterminer l'épaisseur locale de l'écriture depuis une position particulière sur le contour, ce qui n'est pas toujours évident comme le montre la figure 3.42.

Sur les trois directions proposées par la figure 3.42, seules deux seront conservées, celles qui permettent de relier un point de la vallée à un point n'y appartenant pas mais pour lequel il existe un chemin le reliant à l'un des bords de l'image, autrement dit, à un point non inclus dans une boucle. Le segment de coupure doit être le plus vertical possible, la distance entre un point (x^v, y^v) de la vallée et un point de l'extérieur (x^e, y^e) est donnée par :

$$d\left(\binom{x^y}{y^v}, \binom{x^e}{y^e}\right) = |x^v - x^e| + \mu\ |y^v - y^e| \tag{3.22}$$

Le nombre $e_t\binom{x^y}{y^v}$ défini dans l'équation 3.21 est alors égal à :

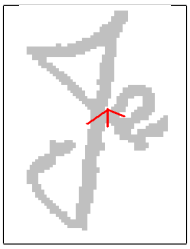


Fig. 3.42: Cette figure illustre une vallée qui propose trois directions différentes de coupure afin de segmenter ce graphème en deux composantes connexes. La première direction (vers la gauche n'est pas adaptée, la direction verticale est souvent la meilleure, la dernière (la plus à droite) mène vers une boucle. Ces trois directions correspondent à trois mesures de l'épaisseur du tracé.

$$e_t\binom{x^y}{y^v} = \min_{(x^e,y^e)} d\left(\binom{x^y}{y^v}, \binom{x^e}{y^e}\right) \tag{3.23}$$

3.6.3 Détection des accents

L'étape suivante consiste à extraire les accents et les points de la séquence de graphèmes déjà obtenue afin de les placer dans une autre séquence. L'idée la plus simple utilise une séparation horizontale de l'image représentée par la figure 3.43. Si cette division est possible, la distance entre les deux objets est alors supérieure à $(\zeta_1 e_t)$ et la surface de l'accent est supérieure à $\left(\zeta_2 \frac{\pi}{4} e_t^2\right)$. Dans ce cas, l'objet supérieur sera nettoyé de la séquence de graphèmes et inséré dans la séquence des accents.

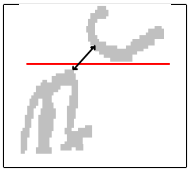


Fig. 3.43: Deux objets séparés par une ligne horizontale. La double flèche représente la distance qui les sépare.

3.6.4 Recollement de petits segments

Il arrive parfois que les méthodes décrites dans les paragraphes 3.6.2 et 3.6.3 divisent les graphèmes de manière trop fine, en particulier en ce qui concerne les traits quasiment horizontaux. Il est parfois utile de recoller de trop petits segments aux lettres voisines afin d'éviter qu'ils ne soient considérés comme des points.

3.6.5 Illustration et résultats

La figure 3.44 présente quelques résultats de cette segmentation obtenue pour les paramètres de la table 3.2 qui ont aussi servi à produire les illustrations intermédiaires. Il subsiste encore des erreurs. L'expérience montre qu'il est impossible d'ajuster les paramètres afin de les faire disparaître sans générer des erreurs sur d'autres documents.

Afin d'évaluer la pertinence d'un traitement dissocié des accents, l'expérience suivante anticipe celle du

paramètre	valeur	équation	page
β	$1,5$	(3.17)	62
τ_1	1	(3.19)	62
τ_2	1	(3.19)	62
τ_3	$0,2$	(3.19)	62
τ_4	$0,3$	(3.19)	62
τ_5	1	(3.21)	64
λ_1	1	(3.20)	63
λ_2	1000	(3.20)	63
λ_3	1	(3.20)	63
λ_4	1	(3.20)	63
λ_5	$0,1$	(3.20)	63
η_1	1	(3.6.2)	64
η_2	$0,5$	(3.6.2)	64
μ	5	(3.22)	66
ζ_1	1	(3.6.3)	67
ζ_2	1	(3.6.3)	67

Tab. 3.2: Liste des paramètres et valeurs utilisés pour la segmentation d'un mot en graphèmes, ces paramètres sont ajustés manuellement à la vue des résultats obtenus sur quelques images prises au hasard dans une large base de données ou sélectionnées de manière automatique en assimilant les graphèmes mal segmentés à des graphèmes peu probables (voir paragraphe 4.3.6, page 103).

paragraphe 4.4 [8]. Elle consiste à comparer les résultats d'une reconnaissance mot, réalisée avec une méthode des plus proches voisins, effectuée sur des images non segmentées dans un premier temps et segmentées en graphèmes dans un second temps. La table 3.3 reprend ces résultats.

L'expérience utilise le jeu de caractéristiques *Mat* décrit au paragraphe paragraphe 4.2.1 car ils sont aussi pertinents sur l'image d'un mot que sur l'image d'un graphème. Tout d'abord, le tableau 3.3 montre que la segmentation fait décroître les performances obtenues pour cette expérience de reconnaissance avec dictionnaire statique, à la fois pour une base d'images de mots anglais et une base d'images de prénoms français. La segmentation peut donc être perçue comme une perte d'information néanmoins nécessaire dans le cas des vocabulaires dynamiques pour lesquels on ne dispose pas d'exemple pour chacun des mots qu'ils contiennent.

Le second résultat concerne trois types de traitements des accents. La première segmentation en graphèmes ne sépare pas les accents comme il est décrit au paragraphe 3.6.3. Le second traitement enlève les accents de la séquence de graphèmes. La troisième option inclut dans la séquence de caractéristiques liées aux graphèmes des caractéristiques décrivant les accents selon le mécanisme décrit au paragraphe 4.3.2. Ces trois traitements aboutissent à des performances similaires sur des bases de mots anglais qui ne contiennent comme accents que des points (sur les lettres "i" et "j"). En revanche, pour une base de prénoms français, le traitement dissocié des accents permet d'accroître légèrement les performances. Toutefois, tenir compte des accents au niveau des caractéristiques ou les oublier ne semble pas faire de différence.

Ces expériences montrent que le traitement des accents n'apporte rien lorsque la langue elle-même n'en contient pas mais il n'altère rien non plus. Pour une langue incluant des accents, il apparaît préférable d'en tenir compte, soit de les nettoyer dans les images où ils apparaissent, soit de les inclure dans les caractéristiques. Les résultats obtenus ne permettent pas de déterminer si une méthode est préférable à une autre. Il reste qu'un traitement dissocié des accents n'est justifié que par leur importance dans la

8. Le paragraphe 4.4 (page 106) précise la source des données ainsi que la manière dont ont été constituées les bases d'apprentissage et de test.

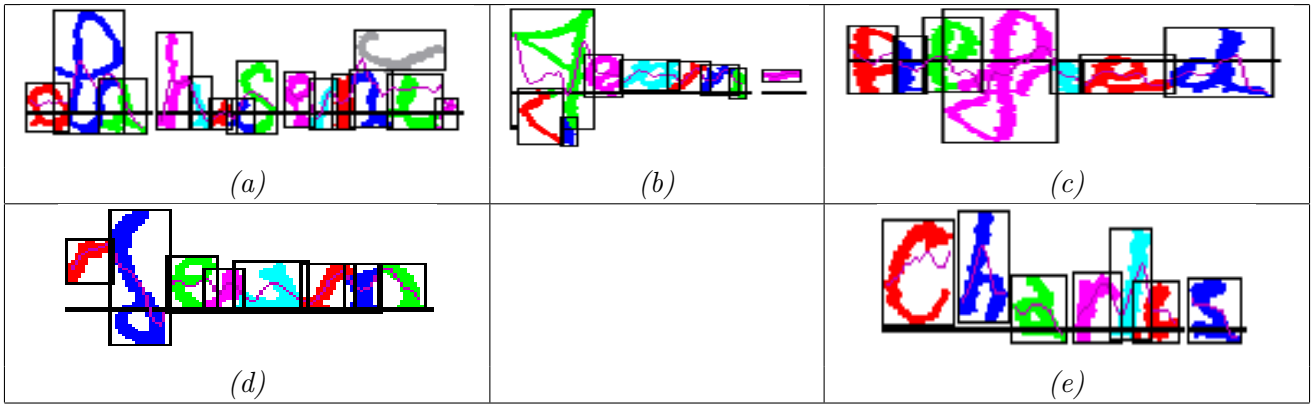


Fig. 3.44: Résultat final de la segmentation graphème. Les valeurs des paramètres utilisées pour cet exemple sont données par le tableau 3.2. Il reste encore des erreurs. Le seul accent segmenté comme tel est celui de l'image "Lahsène".

langue étudiée.

3.6.6 Prolongements

La segmentation en graphèmes proposée ici utilise un grand nombre de seuils de décision (var table 3.2) que l'expérience permet d'ajuster. Au final, le résultat est obtenu après l'application successive d'algorithmes variés de segmentation ou de regroupement. La méthode présentée dans les articles [Desolneux2000], [Desolneux2002], [Desolneux2003] (également abordée au paragraphe 3.4.6) offre une direction de recherche intéressante. Plutôt que de varier les algorithmes, il serait possible de n'utiliser qu'une seule méthode dédiée à la détection de différentes formes géométriques simples telles que les boucles, les ascendants et descendants, les liaisons et autres formes récurrentes de l'écriture. La segmentation s'appuierait sur les frontières des formes détectées. Une telle méthode aurait également l'avantage de ne pas utiliser la connexité entre pixels.

Cette segmentation apparaît comme une multitude de petites recettes appliquées les unes à la suite des autres afin de corriger les imperfections des couches précédentes. Cette première étape, même imparfaite, est néanmoins nécessaire afin de construire une première version des modèles de reconnaissance. Il n'existe pas de définition précise de ce qu'est un graphème mais ce premier jeu de modèles de reconnaissance permet d'extraire les segmentations qui ont participé à une bonne reconnaissance. Il serait possible alors de construire une segmentation en graphèmes apprise à partir de ces bons exemples. La conception d'un tel algorithme est une autre direction de recherche possible pour la poursuite de ces travaux.

3.7 Segmentation en mots

Il est possible de segmenter en mots avant ou après la segmentation en graphèmes. Dans le premier cas, la segmentation est semblable à un découpage en lignes et utilise des projections de l'image selon une direction verticale. Seuls les seuils sont différents. Dans le cas d'une segmentation en mots s'appuyant sur celle en graphèmes, il s'agit de déterminer les graphèmes consécutifs qui appartiennent à deux mots différents.

S'il existe des bases de données contenant des images de lignes déjà segmentées en mots, la seconde méthode utilisant les graphèmes est mieux adaptée. Par exemple, la figure 3.26 contient 17 graphèmes, soit au plus

base	expérience	jeu	taux de reconnaissance
ICDAR	non segmentée	$Mat\,(10,5)$	68,20 %
	non segmentée	$Mat\,(20,10)$	69,45 %
	segmentée (accents inclus)	$Mat\,(5,5)$	52,12 %
	segmentée (accents séparés)	$Mat\,(5,5)$	52,06 %
	segmentée (accents séparés + dist)	$Mat\,(5,5)$	52,07 %
prénoms français	non segmentée	$Mat\,(10,5)$	48,16 %
	non segmentée	$Mat\,(20,10)$	57,59 %
	segmentée (accents inclus)	$Mat\,(5,5)$	40,21 %
	segmentée (accents séparés)	$Mat\,(5,5)$	42,04 %
	segmentée (accents séparés + dist)	$Mat\,(5,5)$	41,96 %

Tab. 3.3: Taux de reconnaissance pour une reconnaissance de mot à l'aide de plus proches voisins. Les bases d'apprentissage et de tests contiennent chacune 15000 mots anglais cursifs appartenant à un vocabulaire de 116 mots différents pour la base ICDAR. Elles contiennent également 15000 prénoms français cursifs parmi une liste de 157 pour la base des prénoms français dont 13,3% contiennent des accents. Les bases d'apprentissage et de test contiennent chacune au moins plus de 100 occurrences d'un mot pour la base ICDAR et au moins plus de 50 occurrences pour la base des prénoms français. Chaque exemple de la base d'apprentissage est classé selon les plus proches voisins dans la base d'apprentissage. Ces voisins sont recherchés à partir d'une distance calculée sur l'image non segmentée ou segmentée.

seize coupures entre deux mots. Le principe consiste à affecter à chacune de ces coupures une probabilité de séparer deux mots, celle-ci est apprise à partir de la base de données et dépend de paramètres tels que la distance entre les deux graphèmes qui l'entourent, leurs tailles, leurs formes... S'il y a N graphèmes, on obtient $N-1$ probabilités de césure $(p_1,\ldots,p_{N-1})$. A chaque point de césure, on associe la variable aléatoire $Y_i \in \{0,1\}$ vérifiant $\mathbb{P}\,(Y_i=1)=p_i$. Une segmentation en mots est alors complètement décrite par la donnée de $(Y_1,\ldots,Y_{N-1})$. Comme ces variables aléatoires sont indépendantes, la probabilité associée à cette segmentation est :

$$\mathbb{P}\,(Y_1,\ldots,Y_{N-1}) = \prod_{i=1}^{N-1} p_i\,(\theta)^{Y_i}\;(1-p_i\,(\theta))^{1-Y_i} \tag{3.24}$$

Chaque $p_i\,(\theta)$ est fonction à valeur dans $[0,1]$ et qui dépend de caractéristiques θ extraites de l'image. Cette fonction peut être par exemple un réseau de neurones [9] estimé en maximisant la vraisemblance (3.24) par rapport à θ sur une base d'images pour laquelle les valeurs $(Y_i)_i$ sont connues. Une fois cette fonction apprise, cette écriture permet de trouver la segmentation en mots la plus probable. Il est également parfois utile de conserver les segmentations les plus probables lorsque l'écriture à découper est ambiguë.

3.8 Post-traitement des graphèmes

Avant de pouvoir reconnaître un graphème ou un caractère, il faut décrire son image à l'aide de caractéristiques qui sont généralement un vecteur de $\mathbb{R}^n$ où n est le nombre de caractéristiques (voir paragraphe 4.2, page 80). Les graphèmes sont parfois très bruités et ce bruit se répercute sur la qualité de leur description. Diminuer l'importance de ce bruit peut améliorer les performances de reconnaissance (voir paragraphe 3.8.1). Ces graphèmes peuvent également inclure plusieurs composantes connexes qui nuisent à certaines extractions de caractéristiques basées sur le contour de la forme (voir paragraphe 3.8.2).

9. Annexes : voir paragraphe C, page 190

3.8.1 Restauration de l'image des graphèmes

Les caractères manuscrits sont parfois mal scannérisés, la binarisation de l'image aboutit parfois à des caractères bruités qu'il est préférable de restaurer. L'article [Whichello1996] se penche sur un bruit diffus qui se manifeste par la dissémination de pixels blancs à travers le caractère à reconnaître (voir figure 3.45). La squelettisation et en particulier l'extraction de contour d'une telle forme est impossible et mène souvent à myriade de petits morceaux proches les uns des autres. La restauration proposée dans [Whichello1996] s'intéresse à l'extraction du contour de la forme bruitée.

Fig. 3.45: Lettre "m" bruitée, la binarisation a conservé environ un pixel noir sur deux.

La méthode s'appuie sur des masques dits (N, M), à partir d'un pixel du contour, on cherche le pixel suivant de ce contour non plus sur un voisinage $(3, 3)$ comme c'est le cas pour une composante connexe mais sur un voisinage (N, M). La table 3.4 illustre les masques $(1, 1)$ et $(3, 3)$. En partant d'un premier pixel, le pixel suivant est alors le pixel noir dont le numéro est le plus faible. Le masque est ensuite tourné selon la direction du déplacement précédemment trouvé.

4	3	2
5	0	1
6	7	8

masque $(1, 1)$

19	18	16	13	12	10	7
22	20	17	14	11	8	6
24	23	21	15	9	5	4
25	26	27	0	3	2	1
28	29	33	39	45	47	48
30	32	35	38	41	44	46
31	34	36	37	40	42	43

masque $(3, 3)$

Tab. 3.4: Masques de différentes tailles pour la recherche du contour, les cases sont numérotées par angle croissant et par distance au centre décroissante. Les autres masques sont obtenus en effectuant des rotations des positions.

L'article [Wang1999] s'attaque à un autre type de détérioration des caractères. La connexité peut être brisée lorsque le caractère dépasse du cadre de l'image ou qu'une partie est escamotée après une binarisation trop rugueuse (voir figure 3.46a). L'algorithme suppose que l'image ne contient qu'une seule composante connexe et cherche à recoller les morceaux si elle en contient plus d'un. Les extrémités du squelette sont prolongées afin d'atteindre une autre composante connexe. Le prolongement est cependant contraint par la courbure du squelette à ses extrémités.

La figure 3.46b permet d'illustrer le coût d'un changement de direction lors du prolongement. A chaque pixel est tout d'abord associée une distance nulle s'il est une extrémité du squelette, infinie dans le cas contraire et un vecteur tangente tenant compte de l'orientation du squelette à son extremité. Cette information est propagée par l'intermédiaire d'une carte de distance [10] utilisant un masque calculé à partir du schéma 3.46b. Les liaisons les moins coûteuses sont conservées de manière à ne former plus qu'une seule composante connexe. Une fois le squelette reconstitué, ce dernier est enrobé d'une épaisseur de pixels conforme à celle du reste de la figure.

L'article [Hwang1998] s'intéresse aux documents imprimés dont les caractères apparaissent en traits trop gras. Les boucles caractères ne sont décelables, noyées par l'épaisseur des traits. Les auteurs utilisent une

10. Annexes : voir paragraphe B.3, page 162

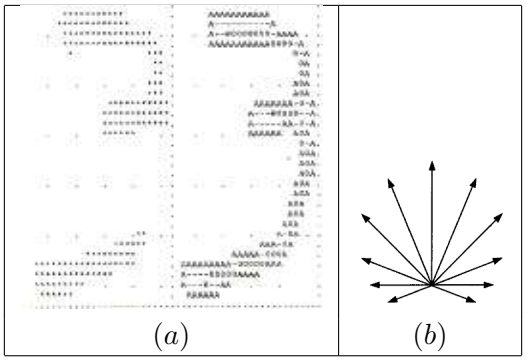


Fig. 3.46: Figure extraite de [Wang1999], les deux chiffres sont incomplets. Les extremités du squelette sont alors prolongées. La figure *b* illustre le coût d'un changement de direction par rapport à une direction verticale.

méthode fondée sur des ondelettes, ces dernières permettant de détecter la présence de segments rectilignes dans une image en niveaux de gris. Cette détection terminée, leur configuration permet de supposer la présence de boucles et ainsi de binariser l'image sans commettre trop d'erreurs (voir figure 3.47).

Fig. 3.47: Figure extraite de [Hwang1998], la première image est l'image originale tandis que la seconde est le résultat du traitement proposé dans [Hwang1998]. Cette binarisation est difficilement accessible aux méthodes reposant sur les simples histogrammes représentant la densité des niveaux de gris (troisième image).

La figure 3.48a montre le dessin d'une lettre "o" partiellement escamotée par la scannerisation. L'œil humain peut facilement reconnaître la lettre "o" même si elle est composée de deux morceaux. Toutefois, la figure 3.48b montre un exemple où il est parfois impossible d'effectuer cette restauration sans avoir connaissance du contexte.

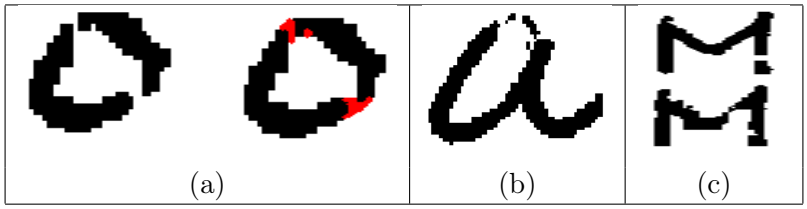


Fig. 3.48: Restauration souhaitée de l'image d'une lettre "o" et restauration ambiguë d'une lettre qui pourrait être soit "a" soit "u". L'image (c) montre le résultat obtenu pour une lettre M et une valeur de α négligeable (voir expression 3.25). La perte de connexité a été corrigée en altérant toutefois le reste de l'image. De petits ergots se sont accrochés sur la partie supérieure de la lettre de façon à créer artificiellement des lignes à trois transitions comme c'est habituellement le cas pour une lettre "M".

A partir d'une classification non supervisée des graphèmes obtenue grâce à un jeu de caractéristiques tels que ceux présentés aux paragraphes 4.2.1 ou 4.2.2, il est possible de déterminer des formes litigieuses, pour lesquelles la classification est ambiguë. Plutôt que de laisser ce doute, la reconnaissance pourrait être améliorée si l'image de départ était modifiée de façon à se rapprocher de l'une des classes avoisinant ce

graphème.

Soit $v(G)$ un vecteur de caractéristiques attaché à un graphème G et $v(H)$ le vecteur attaché au graphème H qui est un exemple représentatif d'une classe quelconque, est-il possible de trouver une forme G' obtenue par une transformation f de coût c_f telle que :

$$d\left(v\left(G'\right), v(H)\right) + c_f \leqslant d\left(v(G), v(H)\right)$$

G est une image dont il est possible d'extraire le contour. A partir de celui-ci, on construit une carte de distance D_G selon la méthode utilisée en annexe [11], cette carte contient pour chaque pixel la distance au pixel noir le plus proche. Pour $\alpha > 0$, la forme G^*_α restaurée est celle qui permet d'atteindre le minimum suivant G^*_α :

$$G^*_\alpha \in \arg\min_{G'} \left[d\left(v\left(G'\right), v(H)\right) + \alpha \sum_{x,y} \left|G'(x,y) - G(x,y)\right| \; D_G(x,y) \right] \tag{3.25}$$

Les différences entre G^*_α et G sont pondérées par leur éloignement par rapport au contour de la forme initiale. Il reste à ajuster α de telle sorte que la restauration ne soit pas trop éloignée de la forme d'origine ni trop discrète. Le meilleur moyen de mesurer l'apport d'une telle méthode est de comparer les performances en reconnaissance entre l'image non restaurée et l'image restaurée. Il est également possible de changer l'équation (3.25) en (3.26) :

$$G^*_\alpha \in \arg\min_{G'} \left[f\left(v\left(G'\right)\right) + \alpha \sum_{x,y} \left|G'(x,y) - G(x,y)\right| \; D_G(x,y) \right] \tag{3.26}$$

avec $f\left(v\left(G'\right)\right)$ densité du vecteur $v\left(G'\right)$ (voir paragraphe 4.3.6)

En supposant raisonnablement que la forme G^* doit rester homotope [12] à G, il est possible de réduire la complexité lors de la recherche du minimum des équations (3.25) et (3.26) en classant les pixels par ordre

11. Annexes : voir paragraphe B.3.1, page 162
12. Annexes : voir paragraphe B, page 159

croissant de distance $D_G(x,y)$. Ceci aboutit à l'algorithme approché suivant :

Algorithme 3.8.1 : restauration
Soient G et H deux graphèmes, l'objectif est de restaurer G en prenant H comme modèle. Soit $\alpha > 0$. La carte de distance $D_G(x,y)$ est construite à partir de l'image du contour en utilisant l'algorithme B.3.4. On suppose également que $(p_1, \ldots, p_n)$ est une suite de pixels vérifiant :

$$\begin{aligned} \forall i, &\quad G(p_i) \neq H(p_i) \\ \forall (i,j), &\quad i \leqslant j \Longrightarrow D_G(p_i) \leqslant D_G(p_j) \end{aligned}$$

Etape A : initialisation
$G' \longleftarrow G$
$m \longleftarrow d(v(G), v(H))$

Etape B : restauration
pour $i = 1$ **à** n **faire**
$G^t \longleftarrow G'$
$G^t(p_i) \longleftarrow H(p_i)$
$m^t \longleftarrow d\left(v\left(G^t\right), v(H)\right) + \alpha D_G(p_i)$
si $m^t < m$ **alors**
$G' \longleftarrow G^t$
$m \longleftarrow m^t$
fin si
fin pour

Pour tester cet algorithme de restauration, la méthode utilisée s'inspire de celle permettant de sélectionner le meilleur jeu de caractéristiques (voir paragraphe 4.3, page 94). Un premier jeu de caractéristiques est choisi de manière à effectuer une classification non supervisée dont le nombre de classes est choisi d'après le critère de Davies-Bouldin [13]. Un second jeu de caractéristiques est choisi de manière à effectuer une classification par la méthodes des plus proches voisins. Quatre tests sont effectués :

1. Le premier test sert de repère : un caractère non restauré de la base de test est classé par rapport à ses voisins non restaurés dans la base d'apprentissage. Ce test est nommé $App\,Test$.
2. Le second test est un compromis : un caractère non restauré de la base de test est classé par rapport à ses voisins restaurés dans la base d'apprentissage. Ce test est nommé $App^r\,Test$.
3. Le troisième test est un autre compromis : un caractère restauré de la base de test est classé par rapport à ses voisins non restaurés dans la base d'apprentissage. Ce test est nommé $App\,Test^r$.
4. Le dernier test : un caractère restauré de la base de test est classé par rapport à ses voisins restaurés dans la base d'apprentissage. Ce test est nommé $App^r\,Test^r$.

1^{er} jeu	nombre de classes	$2^{\text{ème}}$ jeu	$App\,Test$	$App^r\,Test$	$App\,Test^r$	$App^r\,Test^r$
$Prof\,(5,5)$	8	$Prof\,(5,5)$	90,88 %	87,84 %	90,58 %	91,79 %

Tab. 3.5: Résultats obtenus concernant la restauration d'images (la désignation du jeu de caractéristiques reprend celle de la figure 4.4, page 96) pour les quatre tests $App\,Test$, $App^r\,Test$, $App\,Test^r$, $App^r\,Test^r$. Ces résultats ont été obtenus avec environ 2000 images dans les bases d'apprentissage et de test et quatre classes de caractères, "M", "N", "U", "V".

Pour chaque test, le taux de reconnaissance est estimé, les résultats de ces quatre tests sont résumés dans la table 3.5. Etant donné les temps de traitements, ces résultats ont été obtenus sur de petites bases

13. Annexes : voir paragraphe H.2.1, page 350

d'apprentissage et de test (2000 images chacune) et quatre classes de caractères à identifier. Les résultats sont meilleurs que pour une reconnaissance ne prenant pas en compte la restauration. Afin d'expliquer ce gain, on dénombre dans chacune des deux bases d'apprentissage restaurée et non restaurée le nombre d'images pour lesquelles les k plus proches voisins appartiennent à la même classe (voir table 3.6). Cette proportion décroît avec k mais reste toujours supérieure pour la base d'images restaurées, la restauration des images sépare mieux les classes.

k	1	2	3	4
base non restaurée	91,3%	88,2%	85,2%	81,7%
base restaurée	93,9%	91,7%	89,1%	88,2%

Tab. 3.6: Nombre d'images dont les k plus proches voisins appartiennent à la même classe. Cette proportion décroît avec k mais reste toujours supérieure pour la base d'images restaurées.

3.8.2 Connexion de plusieurs composantes connexes

Certaines descriptions de graphèmes utilisent des caractéristiques extraites à partir du contour de l'image. Le contour est alors considéré comme une fonction continue : $f : s \in [0,1] \rightarrow \mathbb{R}$ où s est l'abscisse curviligne. Dans le cas des caractéristiques décrites aux paragraphes 4.2.9 et 4.2.10 (pages 90 et 91), la fonction f est la distance du point du contour dont l'abscisse curviligne est s, au centre de gravité de l'image. Cette méthode n'est pas applicable dans le cas où l'image contient plusieurs composantes connexes. Il devient nécessaire de les relier entre elles afin d'extraire un seul contour. Le paragraphe 3.8.1 mentionne l'article [Wang1999] qui recolle les morceaux d'une lettre, la méthode utilisée ici est plus simple mais peut être considérée comme un cas particulier.

L'idée développée ici s'inspire en partie des travaux de [Wang1999]. Une carte des distances [14] est d'abord extraite de l'image I. A chaque pixel (x, y) sont alors associées deux informations :

$pix_I(x,y)$	est le pixel noir le plus proche du pixel (x,y)
$dist_I(x,y) = d((x,y),\ pix_I(x,y))$	est la distance du pixel (x,y) au pixel noir le plus proche

L'objectif consiste à relier deux composantes connexes différentes par une ligne. Ces lignes doivent être les plus courtes possible afin de ne pas trop altérer l'image originale. Par conséquent, on cherche d'abord les pixels voisins dont les prédécesseurs appartiennent à des composantes connexes différentes. Les lignes qui doivent les relier passent nécessairement pas ces points situés à mi-chemin entre deux composantes connexes, il suffit alors de sélectionner ceux pour lesquels la distance $dist_I(x,y)$ est la plus courte. Ceci

14. Annexes : voir paragraphe B.3, page 162

débouche sur l'algorithme suivant :

Algorithme 3.8.2 : connexion de composantes connexes
Les notations sont celles utilisées dans ce paragraphe, à chaque pixel de l'image I est associé le pixel $pix_I(x,y)$ défini plus haut. On suppose également que $C(x,y)$ est l'indice de la composante connexe à laquelle appartient le pixel (x,y). On désigne le voisinage d'un pixel (x,y) par l'ensemble :

$$V(x,y) = \left\{ (x',y') \mid (x',y') \neq (x,y),\ |x'-x| \leqslant 1,\ |y'-y| \leqslant 1 \right\}$$

Etape A : tri de l'ensemble F
$F \longleftarrow \emptyset$
pour chaque $(x,y) \in I$ **faire**
  pour chaque $(x',y') \in V(x,y)$ **faire**
    si $C(pix_I(x,y)) \neq C(pix_I(x',y'))$ **alors**
      $F \longleftarrow F \cup \{(x,y)\}$
    fin si
  fin pour
fin pour

Etape B : connexion des composantes connexes
L'ensemble F est trié, on note $F = (p_1, \ldots, p_M)$, il vérifie :

$$\forall i,\ dist_I(p_i) \leqslant dist_I(p_{i+1})$$

Etape C : choix des pixels sur les frontières
$c \longleftarrow$ le nombre de composantes connexes
$i \longleftarrow 1$
tant que $(c > 1)$ **faire**
  $c_1 \longleftarrow C(pix_I(p_i))$
  si il existe un voisin q **de** p_i **tel que** $C(pix_I(q)) \neq c_1$ **alors**
    $c_2 \longleftarrow C(pix_I(q))$
    On trace la ligne reliant les points $pix_I(p_i)$ et $pix_I(q)$.
    $c \longleftarrow c - 1$
    pour chaque $(x,y) \in I$ **faire**
      si $C(x,y) = c_2$ **alors**
        $C(x,y) \longleftarrow c_1$
      fin si
    fin pour
  fin si
  $i \longleftarrow i + 1$
fin tant que

Un exemple est donné par la figure 3.49. La lettre A est composée de cinq composantes connexes, chacune est reliée à la plus grande d'entre elles. Il est maintenant possible de n'extraire qu'un seul contour de cette image.

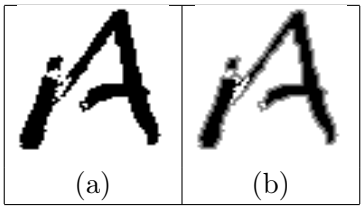


Fig. 3.49: Connexion de composantes connexes, la première image contient quatre composantes connexes dont un pixel isolé. Après leur connexion, il n'en reste plus qu'une : toutes ont été reliées à la plus grande d'entre elles.

3.9 Conclusion

Disposant déjà d'une segmentation en graphèmes [15] fondée sur des heuristiques, l'objectif était d'inclure lors de cette étape une partie apprentissage. Cependant, les méthodes actuelles de segmentations d'image fondées sur des optimisations concernent l'extraction de régions ou la segmentation de textures. La première direction de recherche fut alors l'élaboration d'une segmentation dont les paramètres seraient appris en tenant compte notamment du voisinage des frontières. Mais l'idée développée n'a pas obtenu de résultats satisfaisants.

Ces travaux se sont ensuite orientés vers le problème des accents en langue française qui occasionnent souvent des erreurs de segmentation. A l'aide d'une segmentation en graphèmes inspirés d'algorithmes existants, des expériences ont alors montré qu'un traitement dissocié améliore légèrement les performances en reconnaissance et ne les détériore pas lorsque la langue de l'expérience ne contient pas d'accents. La dernière contribution concerne la restauration des graphèmes abordée ici d'un point de vue statistique, la méthode proposée est lente mais obtient des résultats attrayants à condition de réellement accélérer ce processus.

De plus, cette partie s'est attachée à détailler les prétraitements permettant de décrire l'information contenue dans l'image sous une forme plus exploitable, une séquence de graphèmes, ceux-ci sont ensuite utilisés par les modèles de reconnaissance statistique. Ce processus permet donc de diviser le problème de la reconnaissance d'un paragraphe en une succession de reconnaissances de mots isolés. Il inclut les étapes suivantes :

1. extraction de la zone à reconnaître
2. binarisation
3. nettoyage (soulignement par exemple)
4. correction de l'inclinaison des lignes
5. segmentation en lignes
6. correction de l'inclinaison des lettres
7. segmentation en graphèmes
8. segmentation en mots

La suite concerne la modélisation probabiliste des séquences de graphèmes au travers de modèles de Markov cachés.

15. Celle utilisée dans la thèse [Augustin2001].

Chapitre 4

Reconnaissance statistique

Les traitements d'images aboutissent à une séquence de graphèmes équivalents à des lettres ou des morceaux de lettres. La reconnaissance statistique doit déterminer le mot le plus probable correspondant à cette séquence. Les modèles abordés dans cette partie seront cantonnés aux HMM et IOHMM d'après les conclusions de la première partie (paragraphe 2.6). Les chaînes de Markov cachées permettent de modéliser simplement une séquence discrète, ces modèles sont associés ici à un classifieur continu plus apte à dissocier les différentes formes des graphèmes. Ces modèles hybrides ont été passés en revue dans le paragraphe 2.4.1.

4.1 Préambule

4.1.1 De l'image à la reconnaissance

Ce chapitre dédié à la reconnaissance s'appuie sur les résultats obtenus par le chapitre 3 dédié aux traitements d'images qui détaille la segmentation de l'image d'un mot en une séquence de graphèmes, ce premier résultat est illustré par la figure 3.44, page 69. Chaque graphème est censé représenter une lettre ou un morceau de lettre de sorte que la segmentation en graphèmes est une sous-segmentation de la segmentation en lettres. Ce résultat tant espéré n'est néanmoins pas souvent vérifié comme l'atteste la figure 3.2, page 41. Certains couples de lettres récalcitrants apparaissent régulièrement liés comme les doubles "tt", ou les couples faisant intervenir la lettre "o" comme "œ" ou "oi". Cette mauvaise segmentation intervient dans environ un cas sur dix, ceci implique qu'un mot de plus de dix lettres a toutes les chances de contenir une erreur de segmentation. Bien que non négligeables, ces erreurs ne seront prises en compte qu'au paragraphe 4.7.

Cette segmentation en graphèmes est en fait composée de deux séquences, la première regroupe effectivement des lettres ou des morceaux de lettres, la seconde, de longueur différente ou égale, regroupe des petites images comme les accents ou les points appelés ici les particules aériennes. Ce paragraphe a pour but de présenter ce qui constitue les données sur lesquelles s'appuient les méthodes développées dans ce chapitre et d'exposer succintement l'enchaînement de celles-ci jusqu'au résultat.

La modélisation choisie fait appel aux modèles IOHMM évoqués au paragraphe 2.4.5, page 26 qui prend en entrée deux séquences d'observations. La première appelée séquence d'entrées, sera extraite des liaisons entre graphèmes, de leurs positions relatives. Cette séquence contient autant d'observations qu'il y a de graphèmes et décrit les relations spatiales qui les unissent. La seconde séquence appelée séquence de sortie est extraite directement de la forme des graphèmes ainsi que de la présence des accents. Les particules

aériennes sont en fait considérées comme des prolongements incertains des graphèmes. Chaque graphème est décrit indépendamment des autres et à cette description s'ajoute une moyenne des particules aériennes avoisinantes. La première étape est donc de construire ces deux séquences de liaisons $(L_1, \ldots, L_T)$ et de graphèmes $(O_1, \ldots, O_T)$ illustrées par la figure 4.1 et développées dans les paragraphes 4.2 et 4.3. Ces paragraphes étudient différentes descriptions possibles et leur sélection.

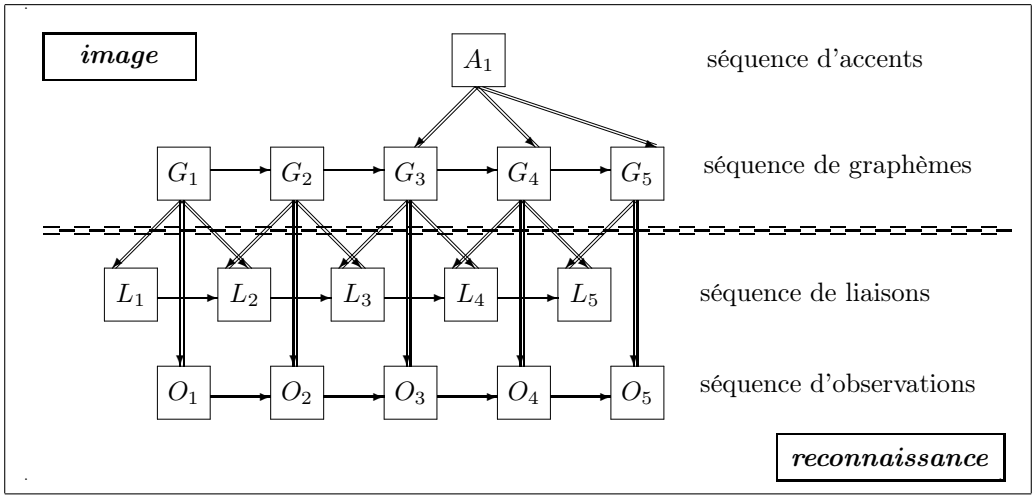


Fig. 4.1: Illustration des séquences de liaisons $(L_1, \ldots, L_T)$ et d'observations $(O_1, \ldots, O_T)$ à partir de la séquence de graphèmes $(G_1, \ldots, G_T)$ et la séquence d'accents ici réduite au seul accent A_1. Les flèches simples indiquent l'enchaînement des éléments à l'intérieur d'une séquence, les flèches doubles indiquent qu'un élément intègre le calcul d'un autre. Les accents peuvent être ainsi reliés par une double flèche aux graphèmes ou directement aux observations.

Une fois ces deux séquences construites, il suffit de leur appliquer les modèles IOHMM afin de procéder à la reconnaissance proprement dite, ce qui sera l'objet des paragraphes 4.5 à 4.9. Ceux-ci s'intéressent aux propriétés des modèles ainsi qu'à leur élaboration. Mais tout d'abord, le paragraphe 4.2 étudiera le passage des graphèmes aux caractéristiques.

4.1.2 Base d'apprentissage et base de test

Le meilleur moyen de jauger les modèles appris par la suite est de vérifier si l'erreur obtenue sur une base ayant servi à l'apprentissage (ou *base d'apprentissage*) est conservée sur une autre base (ou *base de test*) que le modèle découvre pour la première fois.

Soit $B = \{(X_i, Y_i) \mid 1 \leqslant i \leqslant N\}$ l'ensemble des observations disponibles. Cet ensemble est aléatoirement scindé en deux sous-ensembles B_a et B_t de telle sorte que :

$$\begin{array}{l} B_a \neq \emptyset \text{ et } B_t \neq \emptyset \\ B_a \cup B_t = B \text{ et } B_a \cap B_t = \emptyset \\ \frac{card(B_a)}{card(B_t)} = p \in]0,1[\text{, en règle générale, } p \in \left[\frac{1}{2}, \frac{3}{4}\right] \end{array}$$

Ce découpage est valide si tous les exemples de la base B obéissent à la même loi, les deux bases B_a et B_t sont alors dites *homogènes*. Les modèles seront donc appris sur la base d'apprentissage B_a et "testés" sur la base de test B_t. Le test consiste à vérifier que l'erreur sur B_t est sensiblement égale à celle sur B_a, auquel cas on dit que le modèle généralise bien. Le modèle trouvé n'est pour autant le bon modèle mais il est robuste : les résultats obtenus sur une base ne servant pas à l'estimation des coefficients des modèles sont similaires à ceux obtenus sur une base ayant servi à cette estimation. La courbe figure C.14

illustre une définition du modèle optimal comme étant celui qui minimise l'erreur sur la base de test qui ne contient que des "nouvelles" images.

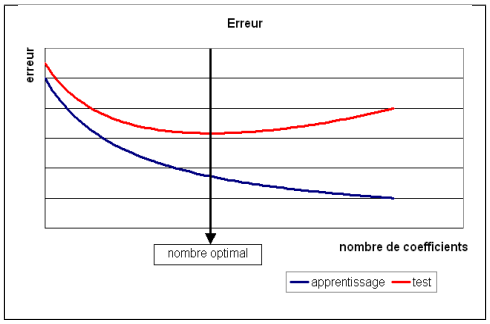


Fig. 4.2: Modèle optimal pour la base de test : les modèles robustes sont situés dans la zone pour laquelle erreur d'apprentissage et erreur de test sont du même ordre de grandeur.

De plus, ce découpage permet de définir une zone valide pour la sélection de modèles située en deça du mimimum obtenu pour la base de test. En dehors de cette zone, les performances obtenues ne peuvent être considérées comme fiables puisque les résultats obtenus sur une nouvelle base d'exemples pourtant de même loi que la base d'apprentissage sont largement supérieurs à ceux obtenus sur une base ayant servi à l'estimation des coefficients des modèles.

4.2 Description des graphèmes

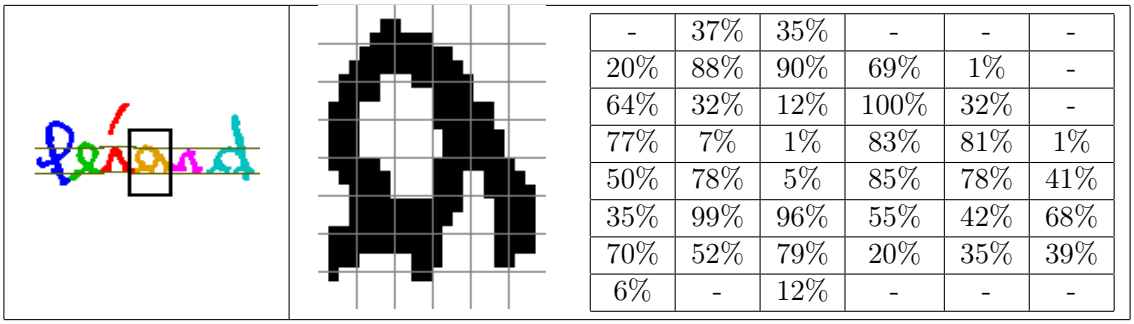
Ce paragraphe reprend plus en détail les caractéristiques présentées au paragraphe 2.2.3. Celles-ci ont pour objectif de traduire une image de lettre ou morceau de lettre dont les dimensions peuvent être variables en un vecteur de dimension fixe et indépendant de la taille du graphème. Ainsi, une séquence de graphèmes sera convertie en une séquence d'observations ou vecteurs de taille fixe. Plusieurs angles d'approche sont présentés, comme des projections des graphèmes sur les axes des abscisses et des ordonnées, une représentation matricielle de l'image, le calcul de moments, diverses caractéristiques extraites du contour... dont les performances seront comparées au paragraphe 4.3.

4.2.1 Matrice

Cette description consiste à décrire des images de tailles différentes par une matrice de réels de dimension fixe. La figure 4.3 illustre ce principe pour une lettre "a" et un découpage en huit lignes et six colonnes. Pour un découpage quelconque en l lignes et c colonnes, afin d'éviter les erreurs d'arrondi, les dimensions (X, Y) du graphème sont multipliées par l en hauteur et c en largeur. Les nouvelles dimensions du graphème sont (Xc, Yl). Si on note $N(x_1, y_1, x_2, y_2)$ le nombre de pixels noirs contenus dans l'ensemble de pixels $\{(x, y) \mid x \in [x_1, x_2] \text{ et } y \in [y_1, y_2]\}$ de l'image agrandie, alors la matrice $M = (m_{ij})_{\substack{1 \leqslant i \leqslant l \\ 1 \leqslant j \leqslant c}}$ des caractéristiques est définie par :

$$\forall (i,j)\,,\ m_{ij} = \frac{N\left(X\left(i-1\right)+1,\ Y\left(j-1\right)+1,\ iX,\ jY\right)}{\max\limits_{i,j} N\left(X\left(i-1\right)+1,\ Y\left(j-1\right)+1,\ iX,\ jY\right)} \tag{4.1}$$

Remarque 4.2.1: minuscule ou majuscule
Si cette matrice suffit à décrire la forme, elle ne permet pas de distinguer un "o" minuscule d'un "O"



-	37%	35%	-	-	-
20%	88%	90%	69%	1%	-
64%	32%	12%	100%	32%	-
77%	7%	1%	83%	81%	1%
50%	78%	5%	85%	78%	41%
35%	99%	96%	55%	42%	68%
70%	52%	79%	20%	35%	39%
6%	-	12%	-	-	-

Fig. 4.3: Représentation d'un graphème sous forme de matrice 8x6 : cette matrice divise l'image en carrés et chacune de ces cases contient le nombre de pixels noirs inclus dans ce carré rapporté au maximum trouvé sur l'ensemble des cases. La première image est celle du mot dont est tirée la lettre "a".

majuscule. Comme toutes les caractéristiques décrites dans ce chapitre, cette distinction n'est possible que si la lettre n'apparaît pas seule.

Il reste à déterminer les dimensions les plus appropriées de la matrice de caractéristiques. Elle ne doit pas être trop petite afin d'être suffisamment discriminante ni trop grande pour éviter une trop grande complexité des modèles de reconnaissance. Ce choix sera débattu au paragraphe 4.3. Bien que cette représentation soit indépendante des problèmes d'échelle, elle est sensible à l'épaisseur des traits ou différentes orientations de caractères (écriture penchée).

On peut supposer que les problèmes d'écriture penchée sont en partie résolus lors d'une étape de correction de l'inclinaison comme celle décrite au paragraphe 3.4.1. L'épaisseur des traits reste tout de même un problème comme l'illustre la figure 4.4 qui pourrait être en partie résolu par l'application de cette matrice sur le squelette des graphèmes ou un squelette d'épaisseur fixée à quelques pixels. Les prétraitements d'images permettent souvent de réduire la variabilité des caractéristiques et ainsi réduire la taille des bases d'apprentissage pour des performances équivalentes.

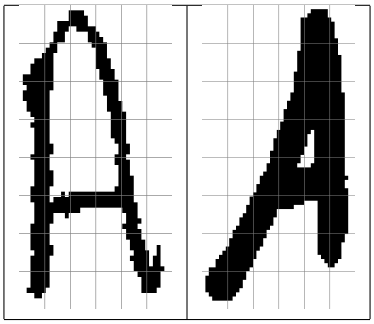


Fig. 4.4: Deux lettres "A" dont les épaisseurs sont différentes et les inclinaisons différentes.

L'article [Oliveira2002] effectue un test entre différents types de caractéristiques dont celles-ci appelées caractéristiques "topologiques" et également celles présentées au paragraphe suivant 4.2.2 qui se révèlent meilleures dans le cadre de l'article.

4.2.2 Profils ou histogrammes

Les profils d'un graphème sont proches d'une projection de celui-ci horizontalement ou verticalement. Sur chaque ligne ou chaque colonne, on peut compter le nombre de pixels noirs qui s'y trouvent ou le nombre de transitions d'un pixel blanc vers un pixel noir équivalent au nombre de traits coupant la ligne. Toutefois, là encore, il s'agit de décrire un graphème selon un vecteur de taille fixe. C'est pourquoi l'image est découpée

en bandes horizontales et verticales et pour chacune d'elles, sont calculées la moyenne du nombre de pixels noirs par ligne (ou colonne) et la moyenne du nombre de transitions par ligne (ou colonne). Un exemple de ce type de description est donné par la table 4.1.

caractéristiques	signification
1..5	épaisseur moyenne sur 5 bandes horizontales / $5\,H$
6..10	épaisseur moyenne sur 5 bandes verticales / $5\,L$
11..15	nombre moyen de transitions sur 5 bandes horizontales / 2
16..20	nombre moyen de transitions sur 5 bandes verticales / 2

Tab. 4.1: Représentation d'un graphème à l'aide de profils pour une image découpée en cinq bandes horizontales et cinq bandes verticales. H et L représentent respectivement la hauteur et la largeur de l'image.

A cet ensemble de caractéristiques, peuvent également être inclus des découpages selon d'autres directions comme les deux diagonales. De plus, il reste encore à choisir la finesse du découpage, c'est-à-dire le nombre de bandes horizontales et verticales le mieux adapté à la reconnaissance. Les questions sont identiques à celles soulevées lors du précédent jeu de caractéristiques (4.2.1) et seront débattues au paragraphe 4.3.

D'autres caractéristiques sont souvent ajoutées à ces profils comme la probabilité que le graphème contienne une boucle, sa surface, sa position. Elles sont succintement décrites dans [Waard1995]. La figure 4.5 illustre une partie de ces caractéristiques.

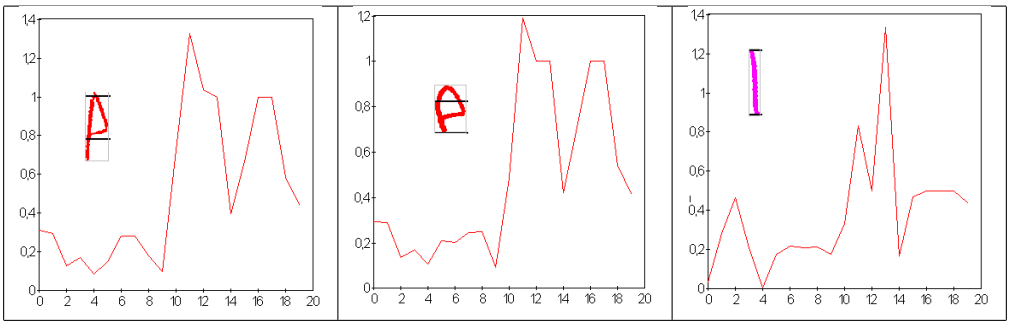


Fig. 4.5: Illustration des projections horizontales et verticales du nombre de pixels noirs regroupées sur cinq bandes dont la description est donnée par la table 4.1. Les deux premiers profils sont sensiblement égaux tandis que le dernier provenant d'une lettre différente ne possède qu'un seul grand pic.

Remarque 4.2.2: pondération des dimensions
Contrairement au jeu de caractéristiques présenté au paragraphe 4.2.1, les informations contenues dans chacune des dimensions de ce vecteur sont hétérogènes. Toutes n'ont pas la même signification et par conséquent, elles évoluent à des échelles différentes. Par exemple, le nombre de transitions est souvent inclus dans l'intervalle $[0, 2]$ alors que l'épaisseur moyenne rapportée à l'image évolue plus souvent dans l'intervalle $\left[0, \frac{1}{2}\right]$. Il est nécessaire de normaliser chaque dimension afin de les rendre homogènes. Les coefficients renormalisateurs peuvent être également appris [1].

4.2.3 Description d'un graphème à partir d'une carte de Kohonen

L'idée est toujours ici de décrire les graphèmes grâce à un vecteur de dimension fixe en plaçant un nombre fixe de points caractéristiques éparpillés dans l'image du graphème. Les cartes de Kohonen permettant de placer ces points (voir [Kohonen1997], [Datta1997]) selon une topologie fixée (linéaire ou quadrillage). La topologie utilisée ici sera un quadrillage contenant quelques dizaines de points. Chaque pixel du graphème

1. Annexes : voir paragraphe H.5.2, page 369

attire vers lui un point du quadrillage de Kohonen qui répercute cette attirance sur ses voisins. Plus en détail, on considère un ensemble de points $(P_{ij})_{\substack{1 \leqslant i \leqslant I \\ 1 \leqslant j \leqslant J}}$ définissant un treillis de Kohonen uniformément réparti sur une image de caractère comme le montre la figure 4.6*a*. Les dimensions de l'image sont notées (X, Y) et la suite (P_{ij}) est initialisée comme suit :

$$\forall\,(i,j)\,,\ P_{ij} = \left(\frac{iX}{I}, \frac{jY}{J}\right)' \tag{4.2}$$

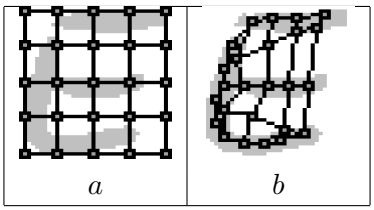


Fig. 4.6: Lettre "E" sur laquelle est superposé un treillis de Kohonen (figure *a*). L'objectif est d'obtenir la convergence des points du treillis vers la configuration de la figure *b* dont on espère qu'elle est propre à la lettre "E".

Les voisins du point P_{ij} sont l'ensemble :

$$V_{ij} = \{P_{i-1,j},\ P_{i,j-1},\ P_{i+1,j},\ P_{i,j+1}\} \bigcap \{P_{ij} \mid 1 \leqslant i \leqslant I \text{ et } 1 \leqslant j \leqslant J\}$$

L'algorithme assurant l'évolution du treillis est le suivant :

Algorithme 4.2.3 : caractéristiques de Kohonen
Etape A : initialisation
$\forall\,(i,j) \in \{1,\ldots,I\} \times \{1,\ldots,J\}\,,\ P_{ij} = \left(\frac{iX}{I}, \frac{jY}{J}\right)'$
$t \leftarrow 0$
$\delta \leftarrow \sqrt{\left(\frac{X}{I}\right)^2 + \left(\frac{Y}{J}\right)^2}$
Etape B : point caractéristique le plus proche et mise à jour
$\alpha \leftarrow \frac{0{,}2}{1+\frac{t}{XY}}$
On tire aléatoirement un pixel p de l'image, si ce pixel p est noir, alors :
$(i^*, j^*) \leftarrow \arg\min_{i,j}\, d\,(P_{ij}, p)$
$P_{i^*,j^*} \leftarrow P_{i^*,j^*} + \alpha\,(p - P_{i^*,j^*})$
Etape C : mise à jour des voisins
$\epsilon \leftarrow \exp\left(\frac{1}{\delta}\,\|P_{i^*,j^*} - p\| - 1\right)$
pour chaque $P \in V_c\,(i^*, j^*)$ **faire**
  $\beta \leftarrow \alpha\,\epsilon\,\exp\left(-\frac{1}{\delta}\,\|P - p\|\right)$
  $P \leftarrow P + \beta\,(p - P)$
fin pour
Etape D : terminaison
$t \leftarrow t + 1$
Tant que l'algorithme n'a pas convergé, retour à l'étape B.

La figure 4.7 illustre le résultat de l'algorithme pour différentes lettres "M", "O", "L", "A", "B", "V". Le vecteur $V \in \mathbb{R}^{2IJ}$ de caractéristiques utilisé pour décrire la forme des lettres provient des coordonnées des

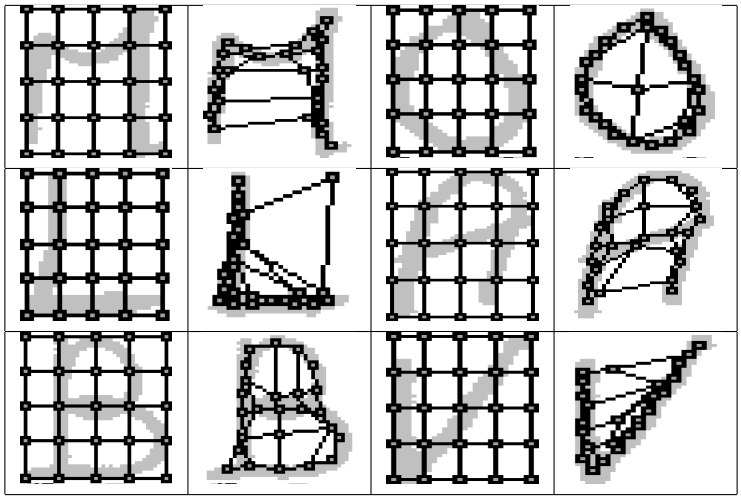


Fig. 4.7: Treillis de Kohonen obtenus pour différentes lettres majuscules.

points du treillis de Kohonen renormalisées afin que les caractéristiques appartiennent à l'ensemble $[0, 1]$. Chaque point du treillis est noté $P_{ij} = \left(p_{ij}^x, p_{ij}^y\right)$, on note $V = (v_1, \ldots, v_{2IJ})$ le vecteur de caractéristiques, les v_i sont définis comme suit :

$$\forall k \in \{1, \ldots, 2IJ\}\,,\ v_k = \begin{cases} \frac{1}{I}\, p^x_{\left[\frac{k+1}{2I}\right], \frac{k+1}{2} \equiv I} & \text{si } k \text{ est impair} \\ \\ \frac{1}{J}\, p^y_{\left[\frac{k}{2I}\right], \frac{k}{2} \equiv I} & \text{si } k \text{ est pair} \end{cases} \tag{4.3}$$

Remarque 4.2.4: ancre
La lettre "O" présente des symétries et il est possible que lors de la convergence, le treillis pivote sur lui-même. Afin d'éviter cela, les quatre coins du treillis peuvent être fixés de manière à orienter le treillis (voir figure 4.8). Cette option n'a pas été testée.

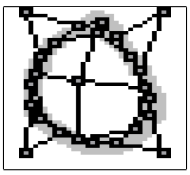


Fig. 4.8: Résultat après convergence du treillis de Kohonen obtenu pour la lettre "O" en maintenant fixes les quatre extremités du treillis.

4.2.4 Description d'un graphème à partir d'un contour

Il est possible de décrire un graphème par son contour mais le résultat obtenu est alors de longueur variable. Le contour est décrit par l'intermédiaire du code de Freeman, en connexité quatre ou huit. Cette séquence peut alors être modélisée par une chaîne de Markov. La figure 4.9 illustre cette description pour deux exemples de lettres "E" et "O". Elle montre que les extremas sont plus marqués pour la lettre "E" que pour la lettre "O". Les barres horizontales sont traduites par les fortes transitions $\rightarrow\rightarrow$ et $\leftarrow\leftarrow$.

Il reste à déterminer la distance entre deux chaînes de Markov modélisant deux contours. La distance adéquate serait une distance de Kullback-Leiber définie pour les deux chaînes de Markov M_1 et M_2. Cette

	→	↑	←	↓
→	0,72	0,17	0,01	0,09
↑	0,27	0,51	0,20	0,02
←	0,01	0,13	0,70	0,16
↓	0,15	0,00	0,30	0,55

	→	↑	←	↓
→	0,25	0,33	0,00	0,41
↑	0,22	0,50	0,28	0,00
←	0,00	0,40	0,32	0,28
↓	0,29	0,00	0,21	0,51

Fig. 4.9: Description de deux lettres à l'aide d'une modélisation du contour à l'aide de chaînes de Markov. Le contour est décrit en connexité quatre et les chaînes de Markov sont du premier ordre, soit seize transitions différentes. Le modèle de la lettre "E" est estimé à partir des contours des deux composantes connexes.

distance n'est pas aisément calculable, par conséquent, les chaînes de Markov seront considérées comme les jeux de caractéristiques et comparées grâce à une distance euclidienne.

Un article récent [Verma2004] utilise également des caractéristiques extraites du contour. Grâce à un système de reconnaissance de l'écriture manuscrite qu'il décrit entièrement, il compare les performances en reconnaissance avec celles obtenues à l'aide d'autres caractéristiques semblables aux histogrammes présentés au paragraphe 4.2.2. Parmi ces caractéristiques figurent entre autres le nombre de changement de direction, de points au dessus et en dessous des lignes d'appui. Les caractéristiques sont multipliées en quadrillant un graphème (comme la méthode du paragraphe 4.2.1) et en étudiant la partie du contour incluse dans chacune des cases.

4.2.5 Seconde description d'un graphème à partir d'un contour

Dans les deux jeux de caractéristiques précédents (4.2.1 ,4.2.2, 4.2.3, 4.2.4), les graphèmes ont été convertis en un vecteur appartenant à un espace vectoriel de dimension fixe. Cette représentation permet de déduire simplement une distance entre graphèmes. Si $\left(V_1^k, \ldots, V_N^k\right)$ pour $k \in \{1,2\}$ sont deux vecteurs associés à deux graphèmes, la distance $d\left(V^1, V^2\right)$ sera définie à partir de la distance euclidienne.

Ces deux représentations décrivent un graphème dans un espace vectoriel et permettent ainsi de leur appliquer des algorithmes classiques de classification. L'objectif de ce paragraphe est de construire une distance entre deux graphèmes reflétant une similitude entre leurs contours sans passer par l'intermédiaire d'un jeu de caractéristiques inclus dans un espace vectoriel de dimension fixe.

La méthode proposée s'inspire des articles [Davies1979], [Tappert1984] ou encore de la thèse [Connell2000] et s'appuie sur le contour des graphèmes. Il ne s'agit plus de décrire un graphème dans un espace vectoriel mais de calculer une distance comparant les contours de deux formes. Pour simplifier, le contour sera réduit au contour extérieur de la lettre et le graphème ne contiendra qu'une seule composante connexe (voir paragraphe 3.8.2).

Soient deux graphèmes G^1 et G^2 et leurs contours C^1 et C^2 décrits grâce au code de Freeman.

$$\forall i \in \{1,2\},\ C^i = \left(c_1^i, \ldots, c_{N^i}^i\right)$$

Les contours sont invariants par rotation, les lettres "p" et "d" ont des contours identiques mais afin de les dissocier, on supposera que l'origine des contours sera leur point le plus haut.

La distance $d\left(C^1, C^2\right)$ entre les deux contours doit être indépendante du facteur d'échelle. Un contour deux fois plus long n'est pas forcément celui d'une lettre différente. Par conséquent, les contours C^1 et C^2 vont être transformés en F^1 et F^2 de même longueur :

$$\left.\begin{array}{rcl} F^1 & = & \left(f_1^1,\ldots,f_{N^{12}}^1\right) = \left(\underbrace{c_1^1,...,c_1^1}_{N^2\ fois},\ c_2^1,c_2^1,...,c_{N^1-1}^1,c_{N^1-1}^1,\ \underbrace{c_{N^1}^1,...,c_{N^1}^1}_{N^2\ fois}\right) \text{ noté } C^1 * N^2 \\ F^2 & = & \left(f_1^2,\ldots,f_{N^{12}}^2\right) = \left(\underbrace{c_1^2,...,c_1^2}_{N^1\ fois},\ c_2^2,c_2^2,...,c_{N^2-1}^2,c_{N^2-1}^2,\ \underbrace{c_{N^2}^2,...,c_{N^2}^2}_{N^1\ fois}\right) \text{ noté } C^2 * N^1 \end{array}\right| \quad (4.4)$$

Les deux contours F^1 et F^2 sont de même longueur $N^{12} = N^1 N^2$. On définit alors la distance $g\left(C^1, C^2, l\right)$ à une rotation près l comme suit :

$$g\left(C^1 * N^2,\ C^2 * N^1,\ l\right) = g\left(F^1, F^2, l\right) = \sum_{k=1}^{N^{12}} \mathbb{1}_{\left\{f_k^1 \neq f_{k+l \equiv N^{12}}^2\right\}} \quad (4.5)$$

Puis, on définit la distance $d\left(C^1, C^2\right)$ entre deux contours par :

$$d\left(C^1, C^2\right) = \frac{1}{N^1 N^2} \min_{1 \leqslant l \leqslant N^{12}} \left\{4 \min\left\{l, N^{12} - l\right\} + g\left(C^1 * N^2,\ C^2 * N^1,\ l\right)\right\} \quad (4.6)$$

Le terme $4 \min\left\{l, N^{12} - l\right\}$ permet de pénaliser la rotation d'un contour par rapport à l'autre. Dans ce cas, une rotation d'un quart de tour est aussi coûteuse que deux contours différant sur chacune de leurs composantes. De manière évidente, la distance (4.6) vérifie :

$$\begin{array}{l} d\left(C^1, C^2\right) \in [0, 2] \\ d\left(C^1, C^2\right) = 0 \Longleftrightarrow C^1 = C^2 \\ d\left(C^1, C^2\right) = d\left(C^2, C^1\right) \end{array}$$

De même, en reprenant la notation (4.4) :

$$\begin{array}{l} \forall k \in \mathbb{N}^*,\ k\, g\left(C^1 * kN^2,\ C^2 * kN^1,\ l\right) = g\left(C^1 * N^2,\ C^2 * N^1, l\right) \\ \Longrightarrow\ \forall k \in \mathbb{N}^*,\ d\left(C^1, C^2\right) \geqslant d\left(C^1 * k,\ C^2\right) = d\left(C^1,\ C^2 * k\right) = d\left(C^1 * k,\ C^2 * k\right) \end{array}$$

Il n'est pas assuré que l'inégalité triangulaire soit vérifiée même si à l fixé, $g\left(F^1, F^2, l\right)$ la vérifie. Soient C^1, C^2, C^3 trois contours, pour tout triplet (i, j, l) tels que $i + j = l$, on a :

$$\begin{array}{rl} g\left(C^1 * N^2N^3,\ C^3 * N^1N^2, l\right) \leqslant & g\left(C^1 * N^2N^3,\ C^2 * N^1N^3, i\right) + \\ & g\left(C^2 * N^1N^2,\ C^3 * N^1N^2, j\right) \end{array}$$

On peut montrer que :

$$\begin{array}{rl} d\left(C^1 * N^2N^3,\ C^3 * N^1N^2\right) \leqslant & d\left(C^1 * N^2N^3,\ C^2 * N^1N^3\right) + \\ & d\left(C^2 * N^1N^3,\ C^3 * N^1N^2\right) \end{array} \quad (4.7)$$

Cette inégalité (4.7) ne montre pas que la fonction $d\left(C^1, C^2\right)$ est une distance. Toutefois, elle s'en rapproche car il est possible de montrer que :

$$\forall k \in \mathbb{N}^*,\ d\left(C^1, C^2\right) \leqslant d\left(C^1 * k,\ C^2 * k\right) + \frac{k-1}{N^1 N^2}$$

Par conséquent, l'inégalité vérifie par d est :

$$d\left(C^1, C^3\right) \leqslant d\left(C^1, C^2\right) + d\left(C^2, C^3\right) + \frac{1}{N^2} \tag{4.8}$$

Bien que la description d'un contour n'appartienne pas à un espace vectoriel de dimension fixe, cette pseudo inégalité triangulaire permet d'envisager l'utilisation d'algorithme de classification tel que celui développé en annexe [2] et utilisé au paragraphe 4.3 permettant la recherche des voisins les plus proches et donc la classification.

Remarque 4.2.5: distance d'édition
La comparaison de deux contours est plus pertinente si la distance (4.5) est remplacée par une distance d'édition [3] comme celle de Levenstein. De cette manière, certains écueils sont évités comme celui de la figure 4.10 où la discrétisation du tracé des lignes peut poser des problèmes.

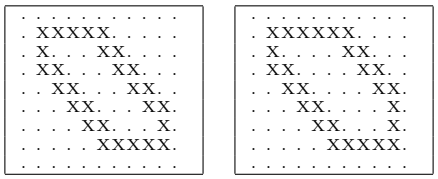


Fig. 4.10: Ces deux trapèzes diffèrent par la longueur de leur base haute et il est préférable d'utiliser une distance d'édition plutôt qu'un simple décalage pour la distance (4.6) qui ne pourra faire correspondre qu'une moitié de ces deux contours qui diffèrent de deux petits déplacements avec une distance d'édition.

4.2.6 Moments invariants

Les étapes de prétraitements d'images incluent fréquemment le redressement des lettres inclinées (voir paragraphe 3.4.1, page 49) qui ressemble presque à une rotation des lettres. Plutôt que d'approfondir dans cette voie, les moments invariants ont été introduits par [Hu1961] puis étendus par [Li1992], [Jin2004], afin de décrire une forme de telle façon que cette description soit invariante par rotation et translation. Ces moments d'ordres différents ne sont pas tous de même dimension ni de même ordre de grandeur, il est par conséquent impossible des utiliser tel quel. L'article [Wong1995] poursuit cette recherche en étendant la liste des moments invariants et en étudiant le facteur renormalisateur obtenant les meilleurs taux de reconnaissance de caractères. L'image est définie par une fonction $f(x, y) \in \{0, 1\}$ qui vaut 1 si le pixel appartient à la forme et 0 dans le cas contraire. L'article définit les moments μ_{pq} par :

2. Annexes : voir paragraphe K, page 390
3. Annexes : voir paragraphe J.2.1, page 386

$$\begin{aligned}
\overline{x} &= \int_x \int_y x\, f(x,y)\ dxdy \\
\overline{y} &= \int_x \int_y y\, f(x,y)\ dxdy \\
\mu_{pq} &= \int_x \int_y (x-\overline{x})^p\ (y-\overline{y})^p\ f(x,y)\ dxdy
\end{aligned} \tag{4.9}$$

Une première liste de moments invariants I_{pq} par rotation est définie dans [Hu1962] :

$$\begin{array}{rcl|rcl}
I_{20} &=& (\mu_{20}-\mu_{02}) - 2i\mu_1 1 & I_{30} &=& \overline{I_{03}} \\
I_{11} &=& \mu_{20}+\mu_{02} & I_{40} &=& (\mu_{40}-6\mu_{22}+\mu_{04}) - 4i\,(\mu_{31}-\mu_{13}) \\
I_{02} &=& \overline{I_{20}} & I_{31} &=& (\mu_{40}-\mu_{04}) - 2i\,(\mu_{31}+\mu_{13}) \\
I_{30} &=& (\mu_{30}-3\mu_{12}) - i\,(3\mu_{21}+\mu_{30}) & I_{22} &=& \mu_{40}+2\mu_{22}+\mu_{04} \\
I_{21} &=& (\mu_{30}+\mu_{12}) - i\,(\mu_{21}+\mu_{03}) & I_{13} &=& \overline{I_{13}} \\
I_{12} &=& \overline{I_{21}} & I_{04} &=& \overline{I_{04}}
\end{array} \tag{4.10}$$

Ces premiers nombres se déduisent de moments invariants calculés en coordonnées polaires. L'article [Wong1995] définit automatiquement d'autres moments invariants par rotation à partir de combinaisons non linéaires à partir des moments définis en (4.10) dont une liste non exhaustive est donnée par la table 4.2. Chaque moment est ensuite renormalisé par le facteur $\frac{1}{\alpha_i}$ où α_i est l'amplitude du moment i.

I_{11}	$I_{20}I_{02}$	
$I_{21}I_{12}$	$I_{30}I_{03}$	$I_{21}^2I_{02}$
$I_{30}I_{12}I_{20}$	$I_{30}I_{12}^3$	$I_{30}I_{21}I_{02}^2$
$I_{30}^2I_{02}^3$	$I_{30}I_{12}^5I_{02}$	
I_{22}	$I_{31}I_{13}$	$I_{40}I_{04}$
$I_{31}I_{02}$	$I_{40}I_{13}^2$	$I_{40}I_{02}^2$
$I_{31}I_{12}^2$	$I_{40}I_{13}I_{02}$	$I_{40}I_{03}I_{12}$
$I_{31}I_{03}I_{21}$	$I_{40}I_{31}I_{03}^2$	$I_{40}I_{13}I_{12}^2$
$I_{40}I_{13}^3I_{20}$	$I_{40}I_{03}^2I_{20}$	...

Tab. 4.2: Moments invariants calculés par la méthode présentée dans [Wong1995]. Ces moments sont complexes mais pour chacun d'eux, les parties réelle et imaginaire (si elles sont non nulles) peuvent être considérées comme caractéristiques.

Le récent article [Heikkilä2004] tente d'apparier une image bruitée à son modèle à partir de ces moments et mesure la sensibilité des caractéristiques choisies par rapport au bruit.

4.2.7 Moments de Zernike

Les moments de Zernike (voir [Trier1996]) sont des projections de l'image d'entrée $f(x,y) \in [0,1]$ sur un espace de fonctions engendré par la suite de fonctions orthogonales $(V_{mn}(x,y))$ définie pour $n \geqslant 0$, $|m| \leqslant n$ et $n-|m|$ est pair :

$$V_{mn}(x,y) = R_{mn}(x,y)\, e^{(i\,m\,\theta(x,y))} \tag{4.11}$$

$$\text{où } R_{mn}(x,y) = \sum_{s=0}^{\frac{n-|m|}{2}} \frac{(-1)^s\ \left(x^2+y^2\right)^{\frac{n}{2}-s}\ (n-s)!}{s!\ \left(\frac{n+|m|}{2}-s\right)!\ \left(\frac{n-|m|}{2}-s\right)!} \tag{4.12}$$

$\theta(x,y)$ correspond à l'angle entre les vecteur $\binom{x}{y}$ et $\binom{1}{0}$. Le moment de Zernike A_{nm} d'ordre n et de répétition m est défini comme :

$$A_{nm} = \frac{n+1}{\pi} \sum_{x^2+y^2 \leqslant 1} f(x,y)\ \overline{V_{nm}(x,y)} \tag{4.13}$$

L'image doit être incluse dans un disque de rayon l'unité mais dans ce cas, l'image originale peut être reconstruite :

$$f(x,y) = \lim_{n\to\infty} \sum_{i=0}^{n} \sum_{\substack{|m|\leqslant n \\ n-|m| \text{ pair}}} A_{nm}\, V_{nm}(x,y) \tag{4.14}$$

En pratique, les coordonnées x et y utilisées ici sont obtenues à partir des coordonnées des pixels en déplaçant l'origine de l'image au centre de gravité puis en changeant son échelle de telle sorte que $x^2+y^2 \leqslant 1$ pour tout pixel.

L'intérêt des moments de Zernike est l'invariance par rotation des modules $|A_{nm}|$ - ils ne dépendent que de x^2+y^2 -. Si celle-ci n'est pas souhaitable (distinction entre les chiffres six et neuf par exemple), il suffit de considérer les parties réelles et imaginaires de ces moments. Comme pour les moments invariants présentés au paragraphe 4.2.6, il est nécessaire de renormaliser les caractéristiques obtenues afin qu'une distance euclidienne puisse être utilisée pour la recherche de plus proches voisins (voir paragraphe 4.3).

4.2.8 Moments estimés à partir d'ondelettes

Plus récemment, l'article [Shen1999] compare les moments de Zernike, de Li et des moments invariants estimés à partir d'ondelettes, sujet de l'article. Les résultats obtenus sont en faveur des moments calculés à partir des ondelettes (100% de reconnaissance), puis ceux de Zernike légèrement moins bons (98,7%) et enfin ceux de Li (75,3%). La base d'ondelettes choisie dans [Shen1999] est celle définie par (4.15) dont les paramètres sont les suivants :

$$p = 3 \quad a = 0,697066 \quad f_0 = 0,409177 \quad \sigma_w^2 = 0,561145$$

L'ondelette est alors définie par :

$$\psi(r) = \frac{4a^{p-1}}{\sqrt{2\pi(p+1)}}\ \sigma_w\ \cos\left[2\pi f_0 (2r-1)\right]\ \exp\left[-\frac{(2r-1)^2}{2\sigma_w^2(p+1)}\right] \tag{4.15}$$

Où $r = \sqrt{x^2+y^2}$, finalement, le moment invariant $F_{m,n,q}$ est défini par :

$$F_{m,n,q} = \frac{1}{\pi} \sum_{x^2,y^2 \leqslant 1} f(x,y)\ e^{i\,q\,\theta(x,y)}\ 2^{\frac{m}{2}}\ \psi\left(2^m r - \frac{1}{2}n\right) \tag{4.16}$$

$\theta(x,y)$ correspond à l'angle entre les vecteur $\binom{x}{y}$ et $\binom{1}{0}$. Ces moments sont calculés pour $m,q \in \{1,2,3,4\}$, et $n \in \left\{0,1,...,2^{m+1}\right\}$.

4.2.9 Profil en coordonnées polaires

Plutôt que de projeter le profil d'une lettre sur l'un des axes du plan, il est possible de calculer une fonction $f : \theta \in [0, 2\pi] \longrightarrow f(\theta)$ en coordonnées polaires ayant pour origine le centre de gravité G de l'image (voir figure 4.11, [Trier1996]).

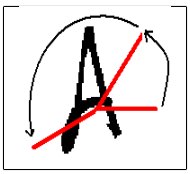


Fig. 4.11: Parcours d'une lettre en tournant autour de son centre de gravité.

La fonction f peut être le nombre de pixels noirs le long de la demi-droite ayant pour origine G et formant un angle θ ayant l'axe des abscisses, elle peut être la distance entre l'origine et le point d'intersection entre cette demi-droite et le contour, elle peut être le nombre de transitions entre pixels blancs et noirs, ces caractéristiques sont les mêmes que celles présentées au paragraphe 4.2.2 à ceci près qu'elles sont estimées sur une droite en rotation par rapport au centre de gravité et non par rapport à une droite parallèle à l'un des axes du plan. Il suffit ensuite de découper l'intervalle $[0, 2\pi]$ en N intervalles identiques pour obtenir un vecteur de N caractéristiques. La figure 4.12 illustre le résultat obtenu, les profils obtenus pour deux lettres "A" sont sensiblement identiques et diffèrent largement de la lettre "I".

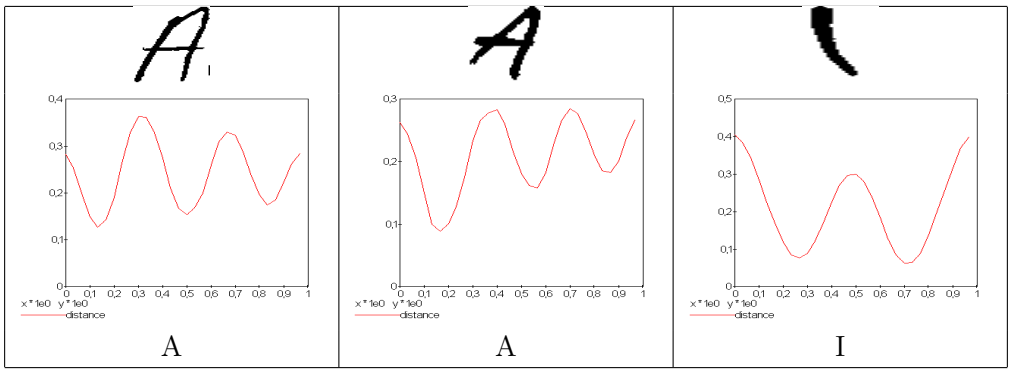


Fig. 4.12: Différents profils de lettres, les deux profils de lettre "A" sont assez proches montrant trois pics, tandis que le profil de la lettre "I" n'en contient que deux.

L'angle θ peut être remplacé par l'abscisse curviligne t le long du contour : $f : t \in [0, T] \longrightarrow f(t) = (x(t), y(t)) \in \mathbb{R}^2$. Le raisonnement est identique à celui qui précède. En revanche, l'image ne doit contenir qu'une seule composante connexe ou s'y ramener en les connectant si elles sont plusieurs (voir paragraphe 3.8.2, page 75).

Toutefois, le problème de l'origine du contour est encore indéterminé. L'article [Wunsch1995] suggère que ce point soit choisi comme étant le plus proche du coin supérieur gauche. Afin d'obtenir des profils comparables, l'origine est choisie comme étant le point le plus proche du coin inférieur droit. Ce choix est arbitraire et peut entraîner de lourdes variations dans le profil obtenu pour le contour, la figure 4.13 illustre deux lettres "F" pour lesquelles les deux profils obtenus seront déphasés. Quel que soit le point de départ, il est vraisemblable que ces cas apparaissent, l'essentiel est que chaque configuration possible ne soit pas un cas isolé ce qui est peu vraisemblable pour de grandes bases de données.

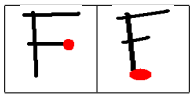


Fig. 4.13: Deux lettres "F" pour lesquelles la règle qui consiste à choisir le point du contour le plus proche du coin inférieur droit retourne deux points différents. Les deux profils seront donc déphasés.

4.2.10 Descripteurs de Fourier du contour

En supposant que l'image à décrire ne soit composée que d'une seule composante connexe, il est possible de considérer le contour extérieur de cette forme comme une fonction $f : t \in [0, T] \longrightarrow f(t) = (x(t), y(t)) \in \mathbb{R}^2$ où t est l'abscisse curviligne le long du contour, T sa longueur totale et $f(t)$ le rayon mesuré depuis le contour jusqu'au centre de gravité de l'image. Si l'image est composée de plusieurs composantes connexes, il est possible de les connecter entre elles afin de n'en avoir plus qu'une seule (voir paragraphe 3.8.2, page 75). L'article [Kuhl1982] (voir également [Trier1996]) propose de construire la transformée de Fourier des deux courbes $x(t)$ et $y(t)$:

$$\hat{x}(t) = a_0 + \sum_{n=1}^{N} \left[a_n \cos \frac{2n\pi t}{T} + b_n \sin \frac{2n\pi t}{T} \right] \tag{4.17}$$

$$\hat{y}(t) = c_0 + \sum_{n=1}^{N} \left[c_n \cos \frac{2n\pi t}{T} + d_n \sin \frac{2n\pi t}{T} \right] \tag{4.18}$$

Les coefficients obtenus seront les caractéristiques qui décriront l'image. Etant donné que le contour extrait est une chaîne de Freeman composée d'une multitude de segments linéaires. Si on suppose que ce contour est décrit en 4-connexité, il est donc composé de T petits segments de longueur 1, le contour balaye les points $(x_t, y_t)_{1 \leqslant t \leqslant T}$ et $(x_0, y_0) = (x_T, y_T)$. La valeur exacte des coefficients est donnée par les formules suivantes :

$$a_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{T} (x_t - x_{t-1}) \left[\cos \frac{2n\pi i}{T} - \cos \frac{2n\pi (i-1)}{T} \right] \tag{4.19}$$

$$b_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{T} (x_t - x_{t-1}) \left[\sin \frac{2n\pi i}{T} - \sin \frac{2n\pi (i-1)}{T} \right] \tag{4.20}$$

$$c_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{T} (y_t - y_{t-1}) \left[\cos \frac{2n\pi i}{T} - \cos \frac{2n\pi (i-1)}{T} \right] \tag{4.21}$$

$$d_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{T} (y_t - y_{t-1}) \left[\sin \frac{2n\pi i}{T} - \sin \frac{2n\pi (i-1)}{T} \right] \tag{4.22}$$

Comme dans la méthode précédente, deux jeux de caractéritisques ne sont comparables que si les deux origines des contours coïncident. On introduit alors un paramètre de phase mesurant le décalage θ_1 de cette origine par rapport à l'axe principal du contour :

$$\theta_1 = \frac{1}{2} \arctan \frac{2(a_1 b_1 + c_1 d_1)}{\sqrt{a_1^2 - b_1^2 + c_1^2 - d_1^2}} \tag{4.23}$$

L'article [Kuhl1982] suggère de placer l'origine du contour au sommet du plus grand axe de l'ellipse, les nouveaux coefficients de la tranformée de Fourier $(a_n, b_n, c_n, d_n)_{n\geqslant 1}$ sont obtenus comme suit :

$$\forall n \geqslant 1, \begin{bmatrix} a_n^* & b_n^* \\ c_n^* & d_n^* \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \begin{bmatrix} \cos n\theta_1 & -\sin n\theta_1 \\ \sin n\theta_1 & \cos n\theta_1 \end{bmatrix} \tag{4.24}$$

Enfin, la forme est tournée de telle sorte que le grand axe de l'ellipse coïncide avec l'axe des abscisses de sorte que les coefficients obtenus seront invariants par rotation. L'axe principal de l'ellipse forme un angle $\theta_2 = \arctan \frac{c_1^*}{a_1^*}$ et :

$$\forall n \geqslant 1, \begin{bmatrix} a_n^{**} & b_n^{**} \\ c_n^{**} & d_n^{**} \end{bmatrix} = \begin{bmatrix} a_n^* & b_n^* \\ c_n^* & d_n^* \end{bmatrix} \begin{bmatrix} \cos n\theta_2 & -\sin n\theta_2 \\ \sin n\theta_2 & \cos n\theta_2 \end{bmatrix} \tag{4.25}$$

Les suites de coefficients $(a_n^{**}, b_n^{**}, c_n^{**}, d_n^{**})$ sont les coefficients désirés.

4.2.11 Autres descriptions

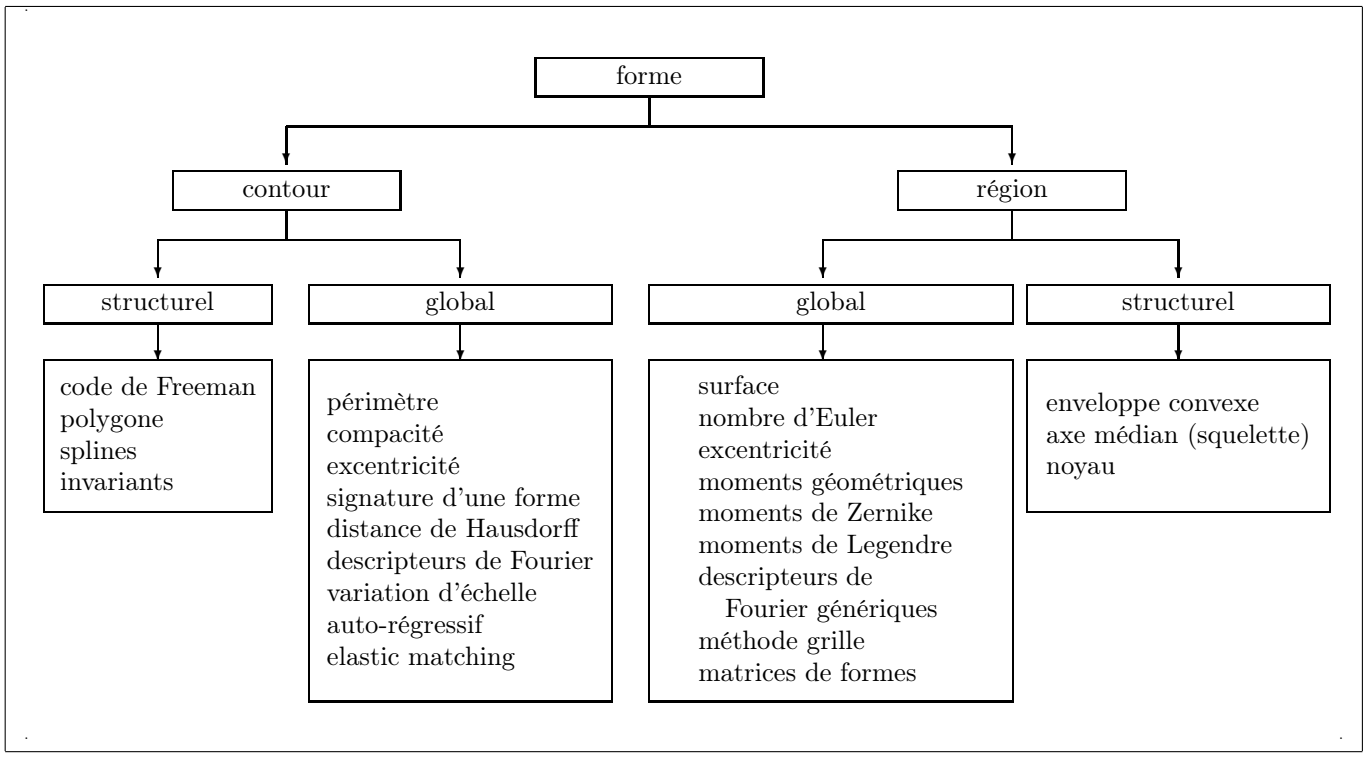


Fig. 4.14: Classification sémantique des représentations (ou descriptions) de formes établie dans l'article [ZhangD2004]. La table 4.3 précise quelques termes utilisés dans ce schéma.

Les descriptions présentées ci-dessus ne sont qu'un échantillon des méthodes utilisées en reconnaissance des formes. Les plus courantes sont passées en revue dans l'article [Trier1996] qui a inspiré certaines des caractéristiques abordées dans ce document. Plus récemment, l'article [ZhangD2004] établit une classification sémantique des représentations de formes, celle-ci est esquissée par la figure 4.14.

Les invariants (premier cadre) regroupent toutes sortes de nombres intrinsèques à une forme et invariants par des transformations telles que les rotations ou les projections. Ces nombres peuvent être des orientations, des rapports de surfaces, de longueurs... L'excentricité est le rapport petit axe sur grand axe. Pour deux ensembles de points $A = \{A_1, \ldots, A_p\}$ et $B = \{B_1, \ldots, B_q\}$, la distance de Hausdorff est définie par : $H(A,B) = \max\{h(A,B), h(B,A)\}$ où $h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\|$. La signature d'une forme correspond au profil en coordonnées polaires (voir paragraphe 4.2.9). Plus généralement, une signature est une courbe à une dimension déduite d'un contour. L'"elastic matching" correspond à une distance égale au coût d'une transformation permettant de passer d'un contour à un autre. Les méthodes auto-régressives envisagent les contours comme des séries temporelles. La variation d'échelle est une description de la forme à plusieurs échelles différentes, semblable à l'introduction de flou. La méthode grille réduit l'image contenant la forme à une grille binaire. Les matrices de forme généralisent la méthode présentée au paragraphe 4.2.1 à toutes sortes de quadrillages plaquées sur l'image, quadrillages rectangulaires, circulaires...

Tab. 4.3: Quelques termes utilisés pour le schéma 4.14, extraits de [ZhangD2004].

Toutefois, parmi ces descriptions, la conception d'un système de reconnaissance fondé sur des modèles de Markov cachés impose une séparation en deux familles : les caractéristiques qui font partie d'un espace vectoriel de dimension finie - par conséquent beaucoup plus simples d'utilisation - et celles dont l'espace de réprésentation est de dimension infinie comme la représentation du contour par une chaîne de Freeman.

Dans la première catégorie, les ondelettes sont souvent utilisées pour décrire des images et sont parfois appliquées aux caractères comme au paragraphe 4.2.8 ou dans l'article [Chen2003]. Il existe des méthodes plus élaborées incluant transformée de Fourier, analyse en composantes principales (ACP) comme dans l'article [Glendinning2003]. Ces méthodes s'appliquent directement sur les graphèmes sans que ceux-ci ne soient prétraités. D'autres utilisent une information plus synthétique comme le contour qui transforme une image à deux dimensions en une fonction de l'abscisse curviligne. Cette fonction peut-être lissée ou traitée comme un signal au moyen de série de Fourier (paragraphe 4.2.10) ou des ondelettes ([Tao2001]).

La seconde catégorie de caractéristiques permet plus de liberté puisqu'il ne s'agit ici que de définir une distance entre caractères. Ce peut être une distance fondée sur le "morphing" d'images (voir [Sederberg1992]) ou ce qu'on appelle "elastic matching" (voir [Uchida2003]). Dans ces deux cas, la distance correspond au coût d'une transformation permettant de transformer une image en une autre.

Cette seconde catégorie inclut également une autre approche complètement différente qui est proposée dans l'article [Khorsheed2003] puisqu'elle est basée sur le squelette plutôt que le contour pour la reconnaissance de mots arabes. Chaque arc du squelette est ensuite vectorisé mais ce découpage d'une image en une multitude de petits morceaux implique une description plus succinte de ces arcs regroupés en un graphe. La description d'un mot n'est plus une séquence de graphèmes mais un graphe de segments. Les travaux développés dans cet article sont toutefois très liés à la structure de l'écriture arabe. Cette description fondée sur le squelette est celle aussi choisie dans l'article [Ruberto2004] et décrite au paragraphe B.7.5, la distance entre deux caractères [4] est alors définie comme une distance entre graphes, eux-mêmes construits à partir du squelette.

Une dernière approche est détaillée dans les articles [Amit1997] et la thèse [Senior1994] qui consiste à

4. L'auteur de l'article [Ruberto2004] n'applique pas sa méthode aux caractères explicitement mais à la reconnaissance des formes en général.

extraire de la base d'apprentissage les caractéristiques les plus fiables. L'article [Amit1997] a déjà été présenté au paragraphe 2.4.9, page 28. La thèse [Senior1994] propose une extraction de caractéristiques à base de contours actifs (ou snakes), ce ne sont pas les seuls mais elles permettent de mieux détecter des formes courantes telles que le "u", "u" inversé, une boucle, un trait vertical. La convergence du contour actif vers une de ces formes complète la décription des graphèmes faite à partir de caractéristiques extraites depuis le squelette de l'image. La forme du contour est comparée à des formes les plus probables grâce à une distance de Malahanobis [5], si cette distance dépasse un certain seuil, le caractère ou graphème en question possède cet attribut de forme.

Toutes ces caractéristiques sont construites depuis l'image et, d'une certaine manière, décrivent un de ses aspects. L'article [Bicego2004] explore la possibilité de construire un ensemble de dissimilarité élaboré à partir de modèles probabilistes, dans ce cas, des modèles de Markov cachés. Le vecteur de caractéristiques utilisés pour décrire une image est cette fois-ci un vecteur de probabilités $(\mathbb{P}(image \mid M_i))_i$. M_i est un modèle de Markov caché ayant appris l'ensemble des images de la classe i ou de préférence un ensemble représentatif de cette classe. Cette méthode propose l'avantage de pouvoir réduire à un seul vecteur tout image ou tout type d'objet comme une séquence d'observations par exemple.

Un article a récemment été publié ([LiuCL2003]) comparant, dans le cadre de la reconnaissance de chiffres manuscrits, les taux de reconnaissance pour divers types de caractéristiques et différents classifieurs tels que les réseaux de neurones, les *support vector machines* ou d'autres modèles reposant sur des apprentissages. Les caractéristiques utilisées sont construites à partir d'images binaires converties en images 20x20 en niveaux de gris. C'est ce que propose de faire le paragraphe suivant : comparer les performances en classification des caractéristiques développées ci-dessus par le biais d'un classifieur basé sur les plus proches voisins.

4.3 Sélection des caractéristiques

4.3.1 Sélection des caractéristiques des graphèmes

Parmi les jeux de caractéristiques proposés (paragraphe 4.2), quelle représentation est la mieux adaptée à la reconnaissance de caractères manuscrits ? L'idéal serait de comparer les performances obtenues en reconnaissance pour l'ensemble du système de reconnaissance. Deux inconvénients majeurs s'opposent à ce procédé, d'une part, ce processus est long car il inclut de nombreux apprentissages (chaînes de Markov cachées, réseaux de neurones) ; d'autre part, les apprentissages doivent être réitérés de nombreuses fois afin que leurs résultats puissent être comparés. En effet, l'apprentissage des réseaux de neurones comme des chaînes de Markov cachées sont effectués grâce à des optimisations utilisant le gradient, ces méthodes convergent vers un minimum de la fonction d'erreur (ici, la vraisemblance des modèles) qui peut être local. Par conséquent, il faudrait répéter plusieurs fois la même phase d'apprentissage afin de s'assurer de la pertinence des comparaisons entre modèles appris sur des jeux de caractéristiques différents. Une autre remarque concerne le nombre de coefficients de ces modèles qui varie forcément puisque les caractéristiques proposées sont de dimensions différentes. Dans ce cas, il faut choisir entre comparer les performances de modèles à coefficients constants et ne considérer que le meilleur modèle, quel que soit le nombre de ses coefficients. Enfin, le jeu de caractéristiques (paragraphe 4.2.5) est une séquence et ne pourrait donc pas être comparé via cette méthode aux autres descriptions de graphèmes qui appartiennent à des espaces vectoriels de dimension finie.

5. La distance de Malahanobis est une distance euclidienne modifiée. Soit X, Y, deux vecteurs de R^n, la distance de Malahanobis entre ces deux vecteurs est définie par : $d(X,Y) = (X-Y)' V^{-1} (X-Y)'$ où V est une matrice de variance covariance. Lorsque la matrice V est estimée sur un nuage de points indépendants et distribués selon une loi normale, cette distance est équivalente à une distance euliclidienne après un changement de répère défini par la matrice $\sqrt{V}$ ($\sqrt{V}$ est définie puisque V est symétrique est donc diagonalisable).

Ces considérations incitent à se diriger vers d'autres méthodes de sélection de caractéristiques, sachant qu'elles sont utilisées dans le cadre d'un problème de classification. Les documents [Fukunaga1990] ou [Choi2003b] proposent des alternatives mais ils supposent néanmoins que ces caractéristiques soient incluses dans un espace vectoriel de dimension finie. La solution retenue s'appuie sur l'algorithme des plus proches voisins et une base d'environ 20000 caractères majuscules (A à Z). La base est divisée en deux parties de même importance. La première constitue la base d'apprentissage (notée B_a), c'est dans cette base que seront cherchés les voisins des éléments de la seconde base intitulée la base de test (notée B_t). Soit $g \in B_t$, $V_a^k(g)$ est l'ensemble de k voisins de g inclus dans la base B_a. La classe d'un élément g est notée $c(g)$ et l'estimation de cette classe à partir du voisanage est notée $\widehat{c(k,g)}$. On définit pour chaque classe i de A à Z et chaque élément g, le coefficient $w(k,g,i)$ défini comme suit :

$$w(k,g,i) = \begin{cases} \infty & \text{si } g \in V_a^k(g) \text{ et } c(g) = i \\ \sum\limits_{h \in V_a^k(g)} \mathbb{1}_{\{c(h)=i\}} \dfrac{1}{d(g,h)} & \text{sinon} \end{cases} \tag{4.26}$$

On définit également un critère :

$$Cr(k,g,i) = \begin{cases} 1 & \text{si } g \in V_a^k(g) \text{ et } c(g) = i \\ 0 & \text{si } g \in V_a^k(g) \text{ et } c(g) \neq i \\ \dfrac{w(k,g,i)}{\sum\limits_{i=A}^{Z} w(k,g,i)} & \text{sinon} \end{cases} \tag{4.27}$$

Finalement :

$$\widehat{c(k,g)} = \underset{A \leqslant i \leqslant Z}{\arg\max}\, Cr(k,g,i) \tag{4.28}$$

Par définition, $Cr(k,g,i) \in [0,1]$ et quel que soit l'élément $g \in B_a$ de la base d'apprentissage et quelle que soit k la taille du voisinage choisie, $\widehat{c(k,g)} = c(g)$. On définit le taux $t(k,B)$ de bonne classification pour une base B contenant N éléments :

$$t(k,B) = \frac{1}{N} \sum_{g \in B} \mathbb{1}_{\left\{\widehat{c(k,g)}=c(g)\right\}} \tag{4.29}$$

D'après ce qui précède, $\forall k > 0$, $t(k,B_a) = 1$. Pour une valeur de k donnée, le meilleur jeu de caractéristiques est celui qui maximise le taux $t(k,B_t)$. Dans la pratique, plusieurs valeurs de k seront utilisées. La table 4.4 rappelle les différents jeux de caractéristiques ainsi que les paramètres qui permettent de les construire.

La table 4.4 précise la distance utilisée pour chaque jeu de caractéristiques. Il n'est pas toujours évident que toutes les dimensions doivent intervenir à poids égal dans le calcul de la distance comme c'est le cas pour une distance euclidienne. Les jeux de caractéristiques faisant intervenir des moments sont renormalisés de sorte que tous les moments aient la même amplitude. L'article [Waard1995] [6] propose une méthode permettant d'apprendre le poids de chaque dimension dans le but d'optimiser les performances en classification. Cette méthode pourrait être employée comme un prolongement de celle présentée dans ce paragraphe.

Caractéristiques	**distance**	**paramètres**	**notations**
matrice § 4.2.1	euclidienne	nombre de divisions horizontales, nombre de divisions verticales	$Mat\,(x, y)$
profils § 4.2.2	euclidienne	nombre de divisions horizontales, nombre de divisions verticales	$Prof\,(x, y)$
carte de Kohonen § 4.2.3	euclidienne	nombre de points sur l'axe des x, nombre de points sur l'axe des y	$Koho\,(x, y)$
chaîne de Markov (contour) § 4.2.4	euclidienne	ordre de la chaîne de Markov (1,2), connexité (4,8)	$Mark\,(d, c)$
contour (séquence) § 4.2.5	édition (4.6)	connexité (4,8)	$Cont\,(c)$
moments invariants § 4.2.6	euclidienne	nombre de moments invariants de Hu	$Hu\,(c)$
profils polaires § 4.2.9	euclidienne	nombre de divisions de l'intervalle $[0, 2\pi]$	$Pol\,(c)$
moments de Zernike § 4.2.7	euclidienne	nombre de moments invariants de Zernike	$Zer\,(c)$
moments avec ondelettes § 4.2.8	euclidienne	nombre de moments invariants à partir d'ondelette	$Ond\,(c)$
descripteurs de Fourier § 4.2.10	euclidienne	nombre de coefficients de Fourier	$Fou\,(c)$

Tab. 4.4: Caractéristiques, distances associées, paramètres relatifs à chaque description de graphèmes, et notations utilisées lors de la présentation des résultats de la sélection du meilleur jeu de caractéristiques.

Afin d'améliorer la rapidité de l'algorithme de recherche des plus proches voisins, la recherche des plus proches voisins est effectuée grâce à l'algorithme LAESA' [7]. Le tableau 4.5 montre la fréquence des caractères utilisés dans la base d'apprentissage, le tableau 4.6 liste les résultats obtenus par cette méthode de classification pour différents jeux de caractéristiques.

Ces résultats montrent dans l'ensemble que les caractéristiques basées sur des moments (Zernike, Li, Fourier) n'obtiennent pas de bons résultats excepté pour les moments basés sur des ondelettes. Aucun de ces jeux ne rivalise avec des caractéristiques de type "géométrique" telles que les jeux Mat, $Prof$, $Cont$, Pol, $BPol$, $Koho$. Les meilleures caractéristiques semblent être celles du jeu $Mat\,(7, 7)$ avec plus de 97 % de bonne classification. Le nombre de voisins ne semble pas avoir beaucoup d'influence, l'écart dépend du type de caractéristiques choisi et n'est pas significatif dans la plupart des cas. Les meilleurs jeux de caractéristiques sont de types Mat, $Prof$, Pol, $BPol$. Ces résultats sont similaires à ceux déduits de l'expérience sur la base MNIST [8].

Il faut noter que le seul jeu non vectoriel $Cont$ permet d'obtenir également de bonnes performances mais le temps de classification reste rédhibitoire (plusieurs secondes par mot) et ne permet d'envisager leur utilisation. Toutefois, il apparaît que l'utilisation du contour "brut" est préférable aux autres descriptions $BPol$ et Fou. Ce résultat suggère que, pour peu qu'on sache constuire une distance entre images, elle serait préférable à toute distance calculée sur des caractéristiques extraites de ces images.

Aucun prétraitement n'a été effectué sur ces bases d'apprentissage utilisées pour définir le voisinage d'une image. Or, parmi ces images, on peut se demander quel sous-ensemble est le plus à même de mener à la

6. Annexes : voir paragraphe H.5.2, page 369
7. Annexes : voir paragraphe K.3.1, page 409
8. base d'images accessible depuis le site *http ://yann.lecun.com/exdb/mnist/*.

A	5,3 %	H	0,2 %	O	5,5 %	V	5,0 %
B	5,5 %	I	0,4 %	P	0,7 %	W	5,7 %
C	5,5 %	J	5,5 %	Q	0,1 %	X	5,8 %
D	5,3 %	K	0,2 %	R	5,4 %	Y	8,7 %
E	5,6 %	L	5,9 %	S	0,4 %	Z	5,6 %
F	5,6 %	M	5,5 %	T	0,5 %		
G	0,24 %	N	5,6 %	U	0,2 %		

Tab. 4.5: Fréquence des caractères dans une base d'apprentissage qui en contient 10000. Cette répartition est identique à celle de la base de test.

jeu	$k=3$	$k=10$	**jeu**	$k=3$	$k=10$
$Prof\,(3,3)$	92,5 %	92,6 %	$Pol\,(10)$	82,3 %	82,9 %
$Prof\,(5,5)$	92,2 %	92,1 %	$Pol\,(20)$	90,1 %	90,0 %
$Prof\,(7,7)$	90,9 %	90,3 %	$Pol\,(30)$	89,5 %	89,3 %
$Mat\,(3,3)$	91,3 %	91,3 %	$Pol\,(50)$	90,7 %	90,4 %
$Mat\,(5,5)$	96,6 %	96,0 %	$BPol\,(20)$	93,2 %	92,8 %
$Mat\,(7,7)$	97,1 %	96,4 %	$BPol\,(30)$	92,8 %	92,3 %
$Koho\,(3,3)$	93,4 %	92,7 %	$BPol\,(50)$	93,6 %	93,2 %
$Koho\,(5,5)$	95,2 %	94,3 %	$Ond\,(10)$	64,3 %	67,0 %
$Zer\,(30)$	54,4 %	59,6 %	$Ond\,(20)$	74,5 %	76,0 %
$Hu\,(30)$	44,7 %	49,2 %	$Ond\,(30)$	79,5 %	80,3 %
$Fou\,(30)$	19,3 %	20,0 %	$Mark\,(1,8)$	-	56,3 %
$Cont\,(30)$	-	97,1 %	$Mark\,(2,4)$	-	62,5 %

Tab. 4.6: Taux de bonne classification $t\,(k, B_t)$ sur la base de test B_t pour différents jeux de caractéristiques et différentes tailles de voisinages. Ces résultats ont été obtenus sur une base d'apprentissage et une base de test de 10000 lettres majuscules.

meilleure classification. Ce choix du meilleur sous-ensemble est relié à la définition de classifiabilité abordée dans l'article [Dong2003]. Cette méthode permettrait d'affiner les résultats présentés dans les tableaux 4.6 et 4.7 voire de rendre possible l'utilisation de classifieurs basés sur des plus proches voisins puisque ceux-ci ont le désavantage d'être coûteux en espace de stockage et en temps lorsqu'il faut déterminer le voisinage d'un point dans un ensemble de plusieurs milliers d'éléments.

Afin de pouvoir utiliser ces graphèmes à partir de modèles de Markov cachés au travers d'un réseau de neurones, il est nécessaire de les classer. La figure 4.15 montre l'évolution de critère de Davies-Bouldin [9] (voir [Davies1979]) en fonction du nombre de classes. Le minimum est obtenu pour 22 classes, toutefois, une dizaine d'autres minima locaux sont assez proches de la valeur obtenue pour le minimum global de la courbe. Le nombre de classes de graphèmes n'est donc pas a priori un paramètre qu'il soit possible de déterminer, il agit simplement sur le nombre de coefficients des modèles.

Remarque 4.3.1: sensibilité au bruit

Il peut être intéressant d'étudier aussi la sensibilité au bruit de différentes caractéristiques. On peut se demander quelle serait la dégradation du taux de reconnaissance en fonction de l'amplitude d'un bruit artificiellement ajouté à l'image. Cette idée permettrait de séparer deux jeux de caractéristiques aux performances équivalentes.

9. Annexes : voir paragraphe H.2.1, page 350

jeu	$k=3$	$k=10$	**jeu**	$k=3$	$k=10$
$Prof\,(3,3)$	85,1 %	86,7 %	$Pol\,(10)$	78,7 %	79,4 %
$Prof\,(5,5)$	89,7 %	90,5 %	$Pol\,(20)$	86,3 %	86,6 %
$Prof\,(7,7)$	90,9 %	91,7 %	$Pol\,(30)$	85,2 %	85,8 %
$Mat\,(3,3)$	67,3 %	70,9 %	$Pol\,(50)$	87,0 %	87,5 %
$Mat\,(5,5)$	86,0 %	87,2 %	$BPol\,(20)$	88,4 %	88,6 %
$Mat\,(7,7)$	94,8 %	94,9 %	$BPol\,(30)$	87,8 %	87,8 %
$Koho\,(3,3)$	85,3 %	92,7 %	$BPol\,(50)$	89,3 %	89,3 %
$Koho\,(5,5)$	92,0 %	92,4 %	$Ond\,(10)$	55,1 %	59,9 %
$Zer\,(30)$	45,8 %		$Ond\,(20)$	60,3 %	63,0 %
$Hu\,(30)$	45,7 %		$Ond\,(30)$	66,9 %	68,9 %
$Fou\,(30)$	43,2 %				

Tab. 4.7: Même expérience effectuée sur la base MNIST contenant 60000 images de chiffres 28x28 pixels pour la base d'apprentissage et 10000 images pour la base de test.

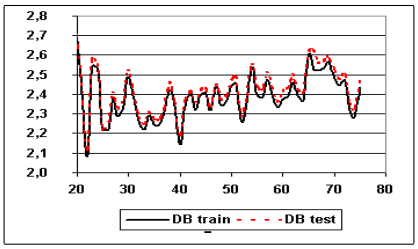


Fig. 4.15: Critère de Davies-Bouldin en fonction du nombre de classes pour les caractéristiques des graphèmes. Cette courbe a été obtenue sur une base d'apprentissage de 400000 graphèmes, une base de test de 130000 images. Les caractéristiques utilisées sont celles notées $Prof\,(5,5)$ (voir table 4.6).

4.3.2 Sélection des caractéristiques des accents

L'écriture d'un mot peut contenir diverses éléments surplombant les lettres comme les accents, les points, les barres de "T" et sont souvent décalés par rapport à leur lettre d'appartenance (voir figure 4.16).

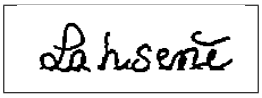


Fig. 4.16: L'accent de la lettre "E" est décalé et surplombe la lettre "N".

Il est possible d'inclure les particules aériennes au graphème qu'elles surplombent mais comme le montre la figure 4.16, ceci peut mener à un résultat erronné. Il est possible également de les considérer comme des graphèmes à part entière et de les insérer dans la séquence de graphèmes, la figure 4.16 montre encore que l'ordonnancement n'est pas évident. Une dernière solution est de juxtaposer la description des accents à celles des graphèmes comme le montre la figure 4.17.

Toutefois, ce procédé a comme inconvénient d'accroître la taille du vecteur de caractéristiques associé à chaque graphème. Etant donné que la variabilité des particules aériennes est moindre que celle des graphèmes, celles-ci seront préalablement classées de manière à convertir un large vecteur de caractéristiques en un vecteur de moindre dimension (moins de cinq). Le nombre de classes obtenu à partir d'une classification non supervisée est déterminé par le critère de Davies-Bouldin [10] (voir [Davies1979]). La figure 4.18

10. Annexes : voir paragraphe H.2.1, page 350

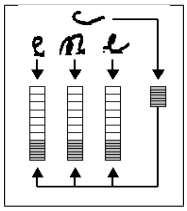


Fig. 4.17: La description de la lettre "E" est ajoutée aux descriptions des graphèmes avoisinants.

montre la courbe obtenue.

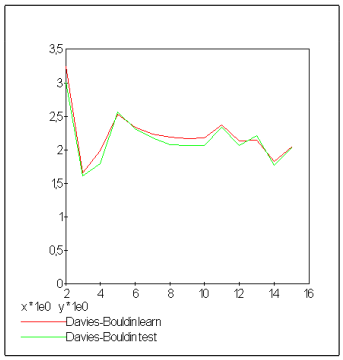


Fig. 4.18: Critère de Davies-Bouldin en fonction du nombre de classes pour les caractéristiques des particules aériennes. Cette courbe a été obtenue sur une base d'apprentissage de 24000 accents, une base de test de 8000 images. Les caractéristiques utilisées sont celles notées $Prof\,(3,3)$ (voir table 4.6).

Le minimum est atteint pour trois classes. Le résultat atteint pour le jeu de caractéristiques $Prof\,(3,3)$ l'est aussi pour le jeu $Mat\,(3,3)$. La table 4.6 montre qu'une trop grande division de l'image nuit à la reconnaissance, c'est pourquoi les dimensions $(3,3)$ ont été choisies comme jeu de caractéristiques pour ces imagettes composées le plus souvent d'un simple trait.

Il reste à mettre en œuvre la relation de voisinage. Un vecteur de caractéristiques V associé à un graphème comporte deux parties $V = (G, A)$ où G est la partie propre au graphème et A celle dédiée aux accents. On note M la suite $M = (m_i)_{i\in\mathbb{Z}}$ définie par :

$$m_i = \frac{2^{2-|i|}}{10}\,\mathbb{1}_{\{|i|\leqslant 2\}} \tag{4.30}$$

Cette suite vérifie $\sum_i m_i = 1$. On considère que chaque accent a est associé au graphème le plus proche $\sigma\,(a)$ et a pour vecteur de probabilités de classification P_a. Pour un graphème d'indice i, A_i la partie décrivant les graphèmes est définie comme suit :

$$A_i = \sum_a m_{i-\sigma(a)}\,P_a \tag{4.31}$$

4.3.3 Sélection des caractéristiques des liaisons entre graphèmes

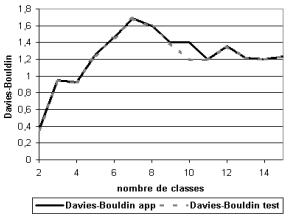
Dans la modélisation qui suit (voir paragraphe 4.5), les liaisons entre graphèmes doivent elles-aussi être décrites par un vecteur de caractéristiques. Aucune annotation n'est disponible pour ces liaisons,

de plus, afin d'éviter l'utilisation de modèles trop lents en terme de temps de calcul, la description de ces liaisons doit être courte. Le tableau 4.8 donne la liste des caractéristiques choisies. Soient deux graphèmes G_1 et G_2, on note leurs boîtes englobantes $r_i = \left(\left(x_1^i, y_1^i\right), \left(x_2^i, y_2^i\right)\right)$ et $r = r^1 \bigcup r^2 = \left((x_1, y_1) = \left(\min\left\{x_1^1, x_1^2\right\}, \min\left\{y_1^1, y_1^2\right\}\right), (x_2, y_2) = \left(\max\left\{x_1^1, x_1^2\right\}, \max\left\{y_1^1, y_1^2\right\}\right)\right)$. $S(r)$ désigne la surface d'un rectangle. Par convention, la première transition entre graphème est celle qui précède le premier graphème, dans ce cas, on note $G_1 = \emptyset$.

indice	**sens**	**expression**	**valeur par défaut ($G_1 = \emptyset$)**
0	rapport des surfaces	$\frac{S(r^1)+S(r^2)}{S(r)}$	1
1	rapport des longueurs	$\frac{x_2^1 - x_1^1}{x_2 - x_1}$	0
2	rapport des longueurs	$\frac{x_2^2 - x_1^2}{x_2 - x_1}$	1
3	rapport des hauteurs	$\frac{y_2^1 - y_1^1}{y_2 - y_1}$	0
4	rapport des hauteurs	$\frac{y_2^2 - y_1^2}{y_2 - y_1}$	1
5	positions relatives	$\frac{x_1^1 - x_1^2}{x_2 - x_1}$	1
6	positions relatives	$\frac{y_1^1 - y_1^2}{y_2 - y_1}$	0

Tab. 4.8: Caractéristiques utilisées pour décrire une liaison (ou transition) entre graphèmes.

Une fois ces caractéristiques désignées, étant donné que les modèles de reconnaissance ont besoin de vecteurs de probabilités, on effectue une classification non supervisée de cet ensemble comme au paragraphe précédent. La figure 4.19 illustre l'évolution du critère de Davies-Bouldin permettant de choisir le nombre de classes approprié puis la figure 4.20 donne quelques exemples pour chaque classe ainsi déterminée.



nb cl : nombre de classes
DB : critère de Davies-Bouldin
app : base d'apprentissage
test : base de test

nb cl	DB app	DB test	nb cl	DB app	DB test
2	0,35	0,35	9	1,41	1,40
3	0,95	0,94	10	1,40	1,19
4	0,92	0,93	11	1,20	1,19
5	1,25	1,24	12	1,35	1,35
6	1,46	1,45	13	1,22	1,22
7	1,69	1,68	14	1,20	1,20
8	1,60	1,60	15	1,23	1,23

Fig. 4.19: Critère de Davies-Bouldin en fonction du nombre de classes pour les caractéristiques décrivant les liaisons entre graphèmes. Le minimum est atteint pour deux classes avec un critère de $0,35$. Le second minimum est atteint pour quatre classes avec un critère égal à $0,92$. Il n'y a pas de différence sensible entre les bases d'apprentissage et de test.

Deux classes permettent d'obtenir un critère de Davies-Bouldin minimal d'après la courbe 4.19. Toutefois, les deux classes ainsi déterminées isolent les liaisons de début de mots des autres. Ce résultat est attendu puisque les liaisons de début de mot sont toutes égales. Par conséquent, cette valeur ne sera pas prise en compte et le bon nombre de classes correspond au second minimum global atteint pour quatre classes. De plus, ce faible nombre de classes convient aux modèles IOHMM développés plus loin (voir paragraphe 4.5.2) car ils évitent un trop grand nombre de coefficients.

Le résultat de la figure 4.20 montre quatre classes et, bien que la classification soit non supervisée, il est cependant possible d'interpréter les regroupements effectués. La seconde classe regroupe les transitions

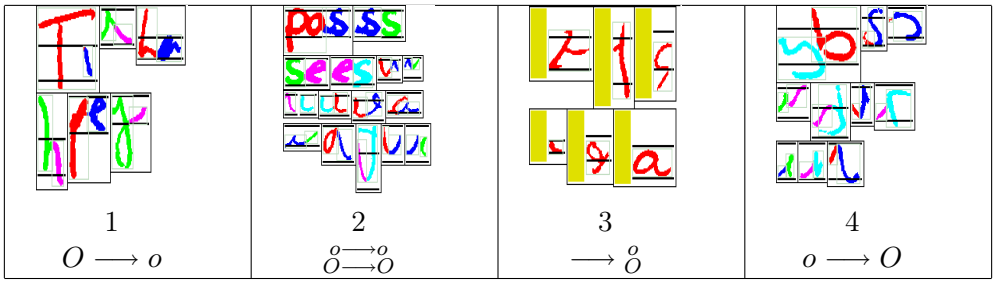


Fig. 4.20: Chaque classe de liaisons entre graphèmes est illustrée par ses représentants les plus probables. o et O représentent respectivement des graphèmes de petite et grande taille. La troisième classe est affectée aux transitions commençant un mot.

entre graphèmes de même importance. La première classe regroupe les transitions entre un grand et un petit graphème, la quatrième, la transition opposée. Quant à la troisième classe, elle regroupe les débuts de mot.

Il est cependant difficile d'évaluer la pertinence des caractéristiques choisies puisque les liaisons ne sont pas identifiables comme le sont les caractères. La classification non supervisée obtenue ci-dessus mène à une configuration qui n'est pas incohérente puisqu'elle aboutit à un résultat interprétable. Néanmoins, il serait préférable d'évaluer les performances en reconnaissance pour différents jeux de caractéristiques en utilisant par exemple la méthode présentée au paragraphe 4.4 qui se contente de mesurer l'apport de cette séquence de liaisons.

4.3.4 Amélioration de la base d'apprentissage

Les tests de classification utilisés pour sélectionner le meilleur jeu de caractéristiques ont été effectués sur une base de données dans laquelle les classes ne sont pas équitablement distribuées. Il y a peu d'exemples pour les lettres rares comme la lettre "W". L'article [Barandela2003] propose de limiter le nombre d'exemples pour les classes sur-représentées. Lorsque la plus petite des classes ne contient que quelques dizaines d'exemples comparés aux milliers que contient la plus grande, cette méthode semble difficilement applicable.

L'autre idée consiste à accroître le nombre d'exemples des classes sous-représentées. La première idée part des images qu'on duplique en y introduisant un bruit comme une érosion, une dilatation de l'image, une rotation de quelques degrés. Il est possible ensuite de bruiter les caractéristiques obtenues pour les rares exemples d'une classe ou d'en fabriquer d'autres à partir de moyennes pondérées.

Soient $(Y_1, \ldots, Y_N)$ les exemples d'une classes, pour obtenir N' autres exemples, on procède comme suit :

$$\forall i \in \{1, \ldots, N'\}, \; Y_i = \sum_{k=1}^{N} \alpha_{ki} \, X_k \tag{4.32}$$

La matrice (α_{ki}) est une matrice aléatoire vérifiant :

$$\forall i, \; \sum_{k=1}^{N} \alpha_{ki} = 1 \text{ et } \forall k, i, \; \alpha_{ki} \sim \mathcal{M}\left(\frac{1}{N}, N\right) \tag{4.33}$$

Pour valider ces hypothèses, un test est construit sur la même base de caractères que celle utilisée au

paragraphe 4.3.1, la base d'apprentissage servant à construire le système de voisinage est améliorée selon deux méthodes possibles :

1. Traitement d'image : multiplication des images par lissage, correction de l'inclinaison, flou (réalisé par l'application d'un filtre (3x3)).
2. Moyenne : les exemples ajoutés sont des moyennes pondérées aléatoirement illustrées par la formule (4.32).

Les tests en reconnaissance sont comparés avec une base d'apprentissage non traitée. Les résultats sont rassemblées dans la table 4.9. L'amélioration que procure la multiplication des images dans la base d'apprentissage par traitement d'image n'est pas significative.

méthode	jeux de caractéristiques	taux de reconnaissance	taux de référence
traitement d'image	$Prof\,(5,5)$	92,16 %	92,07 %
moyenne	$Prof\,(5,5)$	93,68 %	92,07 %

Tab. 4.9: Taux de reconnaissance obtenu pour une base d'apprentissage améliorée comparé à celui obtenu pour une base d'apprentissage non améliorée. La méthode de classification utilise les plus proches voisins (voir paragraphe 4.3.1).

Cette méthode a également été utilisée afin de pouvoir rejeter une partie des images difficiles à classer. Chaque image est présente à six exemplaires après un lissage, une correction de l'inclinaison, et trois images floues obtenues par l'application successive d'un filtre (3x3). Il est alors possible de vérifier pour une image donnée si la classification de ces six exemplaires est concordante auquel cas la classification pour être considérée comme correcte, ceci mène aux résultats de la table 4.10.

jeux de caractéristiques	nombre de concordances	taux de reconnaissance	taux de documents traités	taux de référence	taux de documents traités
$Prof\,(5,5)$	3	92,16 %	100,00 %	92,07 %	100 %
$Prof\,(5,5)$	4	93,13 %	98,11 %	92,07 %	100 %
$Prof\,(5,5)$	5	94,17 %	95,92 %	92,07 %	100 %
$Prof\,(5,5)$	6	96,48 %	89,11 %	92,07 %	100 %

Tab. 4.10: Taux de reconnaissance et taux de rejet en comparant les résultats de classification obtenus pour six exemplaires différents d'une même image modifiée par lissage, rotation, flou.

L'altération des images ne semble pas avoir un impact suffisamment significatif. La seconde expérience montre néanmoins que l'utilisation de plusieurs versions de la même image permet de rejeter certains cas d'erreurs (table 4.10). En revanche, la création artificielle de jeux de caractéristiques comme moyennes des caractéristiques d'une même classe permet d'augmenter le nombre d'exemples pour des classes sous-représentées et aboutit à une amélioration non négligeable des taux de reconnaissance (table 4.9). De plus, cette méthode ne fait qu'accroître la base de données utilisées pour les apprentissages et ne modifie pas la reconnaissance proprement dite.

Ces méthodes supposent de connaître la classe des images à classer et est de ce fait difficilement applicable pour des images contenant des mots. Lors de la reconnaissance, ceux-ci sont segmentés en graphèmes qui ne correspondent pas forcément à des lettres. Néanmoins, si chaque image de mot était segmentée en lettres, il serait alors possible d'augmenter les exemples pour les lettres sous-représentées.

4.3.5 Construction de réseaux de neurones classifieurs

Dans chacun des trois cas présentés ci-dessus (paragraphe 4.3.1 à 4.3.3), le résultat obtenu est un système de classification non supervisée basé sur des centres mobiles. Celui-ci va être converti dans chacun des cas en un réseau de neurones classifieur [11] appris pour retourner le même résultat. L'apprentissage du réseau de neurones est plus adapté au système de reconnaissance qui sera construit. Ceux-ci pourront alors entraîner ce réseau de neurones de sorte que sa classification améliore la reconnaissance. Par conséquent, après maintes itérations d'apprentissage de ce réseau de neurones classifieur, il est possible que la classification qu'il effectue alors diverge de celle obtenue initialement.

Les réseaux de neurones supportent mal les classes sous-représentées, par conséquent, l'apprentissage est effectué sur une base d'apprentissage pour laquelle les exemples sont multipliés artificiellement (comme au paragraphe 4.3.4). Trois réseaux de neurones sont ainsi obtenus, le premier classe les graphèmes, le second les accents, le dernier les liaisons entre graphèmes. Le résultat de la classification des accents sera ajouté au vecteur de caractéristiques des graphèmes selon les expressions (4.30) et (4.31). A chaque image correspond donc deux séquences de caractéristiques illustrées par la figure 4.1.

4.3.6 Valeurs aberrantes

La segmentation en graphèmes, appliquée à des images de mauvaise qualité ou une écriture peu lisible, produit souvent des résultats de mauvaise qualité (figure 4.21). Peu courants, ces graphèmes introduisent un bruit non négligeable qui sera appris par les modèles de reconnaissance. La lettre "e", très courante, n'en sera pas affectée mais la lettre "w", peu représentée en langue française, présente à peine plus d'une centaine de fois dans les bases d'apprentissage, pourrait être mal reconnue par la suite.

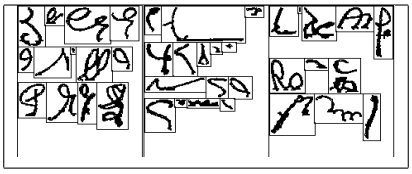


Fig. 4.21: Graphèmes de mauvaise qualité, peu lisibles. Hors contexte, pour la plupart, ils ne semblent faire partie d'aucune lettre.

L'autre objectif est par la suite de pouvoir construire un estimateur de qualité de l'écriture. Plus un mot est lisible, plus sa reconnaissance est susceptible de retourner un résultat correct. Si tous les graphèmes sont lisibles, alors le mot a toutes les chances de l'être aussi. La lisibilité est une notion subjective et difficile à transcrire en termes mathématiques. Toutefois, on peut supposer qu'un graphème est lisible si beaucoup d'autres lui ressemblent, donc si de nombreux autres graphèmes sont proches dans l'espace qui a été choisi pour les représenter. Un estimateur à noyau de la densité permettra de caractériser cette proximité.

Les résultats présentés sont inspirés du livre [Silverman1986] et de l'article [Herbin2001] [12]. L'estimateur à noyau choisi est basé sur un noyau gaussien multidimensionnel et pour un échantillon de variables i.i.d. (identiquement et indépendamment distribuées) $(X_1, \ldots, X_N) \in \left(\mathbb{R}^d\right)^N$, on a :

$$\hat{f}_H(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\det H} K\left(H^{-1}(x - X_i)\right) \quad \text{où} \quad K(x) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{\|x\|^2}{2}} \tag{4.34}$$

11. Annexes : voir paragraphe C.1.5, page 196
12. Annexes : voir paragraphe H.2.2, page 351

L'estimation dépend de la matrice H qui peut dépendre localement de x ou non. Etant donné que N est de l'ordre de 200000 et d de l'ordre de quelques dizaines, la matrice H a été restreinte à sa diagonale $H = diag\,(h_1, \ldots, h_d)$ et est déterminée selon la règle de Silverman :

$$\begin{aligned} \forall j \in \{1, \cdots, d\},\ \hat{h_j} &= \hat{\sigma}_j\, N^{-\frac{1}{d+4}} \\ \text{avec } \hat{\sigma}_j &= \sqrt{\mathbb{V}X_j} \\ &\text{où } X_j \text{ est la variable aléatoire réelle correspondant à la } j^{\text{ième}} \text{ coordonnée} \end{aligned} \tag{4.35}$$

Afin d'éviter les graphèmes les moins fréquents, la suite $\left(X_{\sigma(1)}, \ldots, X_{\sigma(N)}\right)$ est ordonnée de telle sorte que $\hat{f}_H\left(X_{\sigma(1)}\right) < \cdots < \hat{f}_H\left(X_{\sigma(N)}\right)$. Pour $\alpha \in [0,1]$, on définit $q_\alpha = \hat{f}_H\left(X_{\sigma([\alpha N])}\right)$. Pour un graphème quelconque x, on peut supposer que si $\hat{f}_H(x) \leqslant q_{5\%}$, alors le graphème est plutôt mal écrit car il fait partie de l'ensemble des 5% de graphèmes les moins probables. La figure 4.22 montre les courbes $(\alpha, q_\alpha(H_{Sil}))$ obtenues pour les bases d'apprentissage et de test. Ces deux courbes diffèrent complètement et cela signifie que la règle de Silverman ne convient pas sur ce genre de problème. Par conséquent, la seconde règle adoptée est de trouver un vecteur $H(\gamma)$ colinéaire avec H_{Sil} ($H(gamma) = \gamma H_{Sil}$) de telle sorte que les deux courbes coïncident. La quantité $\Delta(\gamma)$ définie en (4.36) décroît presque constamment lorsque γ augmente (voir figure 4.23) pour devenir quasi-nulle lorsque γ est très grand, ce cas correspond à une densité uniforme. Il suffit de choisir la plus petite valeur pour γ de sorte que $\Delta(\gamma)$ soit inférieur à un certain seuil.

$$\Delta(\gamma) = \sum_i \left[q^{app}_{\alpha_i}(H(\gamma)) - q^{test}_{\alpha_i}(H(\gamma))\right]^2 \tag{4.36}$$

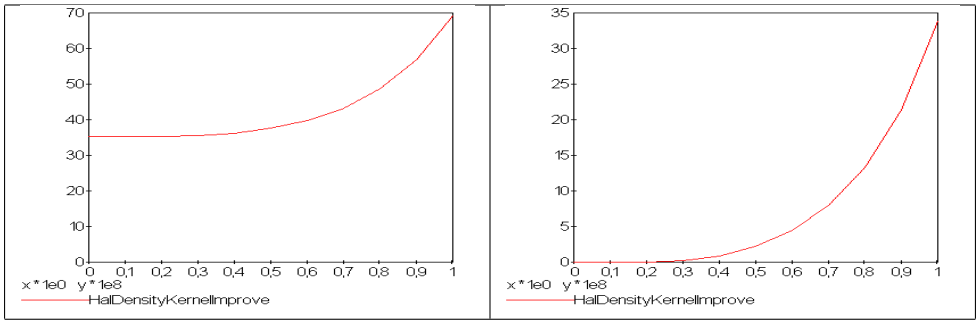


Fig. 4.22: Courbes $(\alpha, q^{app}_\alpha(H_{Sil}))$ et $(\alpha, q^{test}_\alpha(H_{Sil}))$ avec $\alpha \in \left\{\frac{i}{20} \mid 0 \leqslant i \leqslant 20\right\}$ obtenues sur les bases d'apprentissage et de test et en appliquant la règle de Silverman. Leur différence de forme suggère que la règle de Sileverman (4.35) fonctionne mal sur ce type de données.

Cette idée a d'abord été vérifiée sur des données simulées illustrées par la figure 4.23. L'image 4.23a montre l'évolution du critère $\Delta(\gamma)$ en fonction de γ. Dans ce cas, la meilleure valeur pour γ est 1, ce qu'illustre l'image 4.23b pour un mélange de deux lois normales.

La figure 4.24 illustre les résultats obtenus dans le cas réel. La courbe $(\log\gamma, \log\Delta(\gamma))$ est quasiment une droite qui permet de calculer une valeur approximative pour γ égale à $4,18$. Les résultats obtenus à partir de cette valeur sont donnés par la figure 4.25.

L'estimation de la densité permet de révéler les graphèmes très mal segmentés comme le montre la figure 4.25. A l'inverse, les graphèmes les plus probables sont des traits verticaux comme les barres de la lettre u ou celle de la lettre i. Néanmoins cette modélisation est assez coûteuse puisque l'expression (4.34) nécessite une somme sur un grand nombre de données. Il est possible d'optimiser ce calcul en restreignant

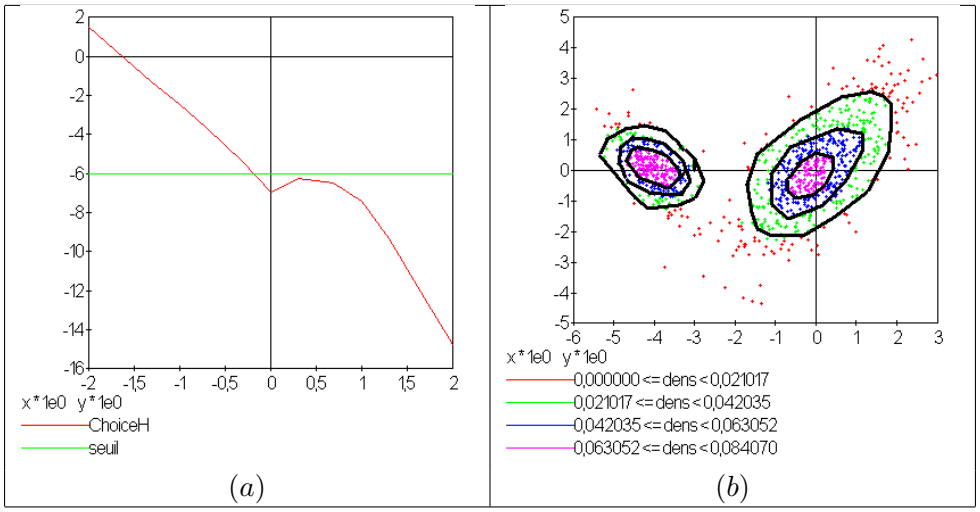
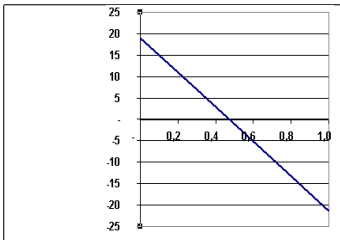


Fig. 4.23: La première image représente la courbe $(\log\gamma,\, \log\Delta(\gamma))$, la seconde courbe représente les quartiles de la variable aléatoire $f_X(X)$ où f_X est l'estimateur à noyau de la variable X sur un échantillon de 1000 points simulés selon un mélange de deux lois normales. Pour cet exemple, la valeur choisie pour γ est 1, ce qui revient à appliquer la règle de Silverman.



$$\log\gamma \approx -40,4\,[\log\Delta(\gamma)] + 19,1$$

$\log\gamma$	$\log\Delta(\gamma)$
0,00	19,05
0,30	7,01
0,70	-9,19
1,00	-21,33

Fig. 4.24: Courbe $(\log\gamma,\, \log\Delta(\gamma))$ observée avec des graphèmes décrits par le jeu de caractéristiques $Prof\,(5,5)$. Pour $\gamma = 4,18$, on trouve $\log\Delta(\gamma) \approx -6$.

la somme uniquement aux voisins les plus proches [13]. Il est aussi possible de compresser cette information en approximant cette densité par un mélange de lois normales multidimensionnelles. Ces densités sont estimées à partir de l'algorithme EM [14] (Expectation Maximization, voir [Dempster1977]) pour lequel le mélange peut inclure un nombre variable de lois normales.

Cette densité a dans un premier temps été utilisée afin de sélectionner de manière automatique les graphèmes mal segmentés afin d'ajuster les paramètres de la segmentation en graphèmes présentée au paragraphe 3.6. Il a également été envisagé d'intégrer ce critère dans l'élaboration d'un critère de confiance associé aux résultats de la reconnaissance. Cet objectif supposait toutefois d'obtenir cette densité par un autre moyen plus rapide qu'un estimateur à noyau comme un réseau de neurones ou un mélange de lois gaussiennes par exemple. Cette direction nécessite le choix d'un modèle, la sélection du nombre de ses coefficients, des directions de recherche qui n'ont pas encore été développées.

La méthode présentée ici s'appuie sur une modélisation non paramétrique de la densité plutôt qu'une modélisation paramétrique comme par exemple un mélange de lois gaussiennes. L'article [Hoti2004] présente

13. Annexes : voir paragraphe K, page 390
14. Annexes : voir paragraphe H.4.1, page 359

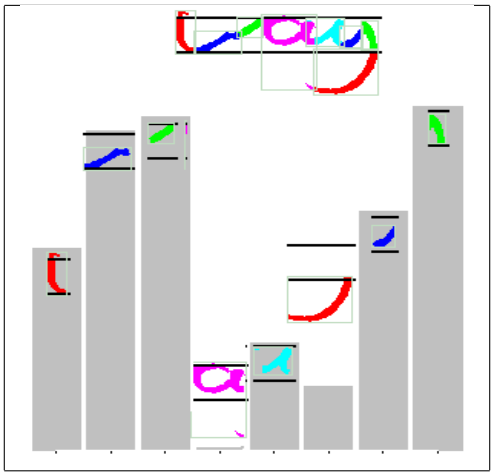


Fig. 4.25: Densité obtenue pour les graphèmes suivants, selon le jeu de caractéristiques $Prof\,(5,5)$ décrits aux paragraphes 4.2.1 et 4.2.2. Le quatrième graphème possède une densité qui l'inclut dans les 5% de graphèmes très peu probables, la lettre a est en effet accompagnée d'un morceau de la lettre y.

une modélisation semi-paramétrique [15] qui pourrait également être étudiée. La densité du couple (x, y) est alors estimée par : $f_{x,y}(x,y) = f_{y\,|\,x}(y)\, f_x(x)$. La densité $f_x\,(x)$ est estimée de façon non paramétrique tandis que $f_{y|x}\,(y)$ est modélisée par une loi gaussienne. x et y pourraient par exemple être deux jeux de caractéristiques différents avec x un vecteur de petite dimension (moins d'une dizaine). Plus la dimension du vecteur x est faible, plus le calcul de la densité en un point est rapide.

4.4 Reconnaissance de mots cursifs à l'aide de plus proches voisins

Avant de commencer la description de modèles complexes associant chaînes de Markov et réseau de neurones, un premier système de reconnaissance est implémenté, basé sur un système de classification fondé sur les plus proches voisins. Chaque image est traduite par une séquence d'observations de longueur variable. Pour mettre en œuvre cette idée, il suffit de construire une distance entre séquences d'observations.

4.4.1 Distance d'édition

Soient deux images de mots dont sont extraites deux séquences d'observations $O^1 = \left(O^1_1, \ldots, O^1_{T_1}\right)$ et $O^2 = \left(O^2_1, \ldots, O^2_{T_2}\right)$, on cherche à définir une distance $d\left(O^1, O^2\right)$ qui traduise la proximité de ces deux mots. Il est possible de construire cette distance comme une distance d'édition [16] où le coût de comparaison entre deux observations est estimé par une distance euclidienne. Il reste donc à déterminer le coût d'insertion ou de suppression d'une observation.

On suppose que X et Y sont deux variables indépendantes de même loi normale $\mathcal{N}\left(\mu, \sigma^2\right)$. La variable $X - Y$ suit une loi normale de paramètre $\mathcal{N}\left(0, 2\sigma^2\right)$. Par conséquent, $\frac{1}{2\sigma^2}\,(X-Y)^2$ suit une loi χ_2 dont on déduit que :

$$\mathbb{P}\left(\frac{1}{2\sigma^2}\,(X-Y)^2 \leqslant 0,455\right) \sim 0,5$$

D'une manière grossière, on suppose que les observations sont des vecteurs dont chaque dimension suit

15. Annexes : voir paragraphe H.5.4, page 371
16. Annexes : voir paragraphe J, page 382

une loi normale et sont indépendantes les unes des autres. Soit $\left(O^1, \ldots, O^N\right)$ l'ensemble des observations de la base d'apprentissage, on désigne par V_i la variance de la i$^{\text{ème}}$ dimension :

$$\forall i,\ V_i = \frac{1}{N}\sum_{k=1}^{N}\left(O_i^k\right)^2 - \left[\frac{1}{N}\sum_{k=1}^{N}O_i^k\right]^2 \tag{4.37}$$

Le coût d'insertion et de suppression c est défini par :

$$c = 0,455 \times 2\sum_i V_i \sim \sum_i V_i \tag{4.38}$$

Si les observations vérifient les hypothèses d'indépendance et de normalité - ce qui est rarement le cas -, soient deux observations O^1 et O^2, alors :

$$\mathbb{P}\left(\left\|O^1 - O^2\right\|^2 \leqslant c\right) \sim 0,5$$

La figure 4.26 donne le calcul des distances pour deux groupes de deux images correspondant au même mot. Dans ce cas, les mots appartenant à la même classe forment un couple de plus proches voisins. Il reste à évaluer cette idée à grande échelle.

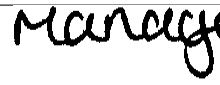
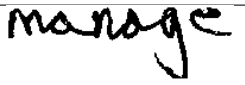
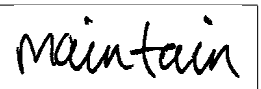
images indices	 (1)	 (2)	 (3)	 (4)
$d((1),.)$	-	9	27	35
$d((2),.)$	9	-	26	33
$d((3),.)$	27	26	-	16
$d((4),.)$	35	33	16	-

Fig. 4.26: Quatre images de mots et leurs distances respectives selon la distance décrite au paragraphe 4.4.1. La distance entre deux mots identiques n'excède pas 20, la distance entre deux mots différents est aussi supérieure à ce seuil.

4.4.2 Résultats

Trois expériences sont réalisées afin d'observer les performances en reconnaissance d'un tel système. La première (ICDAR1) utilise un ensemble de 1600 mots anglais, chacun est présent dans la base d'images au moins dans quatre d'entre elles et dans au plus cent images. La seconde (ICDAR2) expérience utilise un ensemble de 116 mots anglais, leur nombre d'occurrences est compris entre 100 et 743 (voir figure 4.27). Dans les deux cas, la base d'images contient environ 25000 images de mots anglais [17] (voir [ICDAR2003]).

La troisième expérience (PRENOMS) utilise un ensemble de 157 prénoms français non composés, leur nombre d'occurrences est compris entre 50 et 3400 environ [18]. La base d'images contient 30000 prénoms.

17. Cette base est extraite de celle donnée à l'adresse *http ://www.cs.nott.ac.uk/∼dge/words.tar.gz* sur le site ICDAR 2003 (voir [ICDAR2003]). Les mots ont été ordonnés par ordre d'occurrence décroissante, ceux dont la fréquence est inférieure à 4 ou supérieure à 743 ont été éliminés afin de conserver 50000 mots répartis sur deux expériences. Les mots les plus fréquents qui ont été rejetés sont les suivants : with, do, was, not, as, me, this, am, it, in, that, my, of, a, and, the, to, i, ce sont de petits mots qu'il n'est en général pas utile de reconnaître. Les mots les moins fréquents sont de grands mots et très significatifs (antidepressants, astrological, backgammon...).

18. Quelques-uns des prénoms moins courants sont "Clémence", "Aimée", "Nelly", les plus courants sont "Pierre", "Louis", et de loin les plus courants, "Jean" et "Marie".

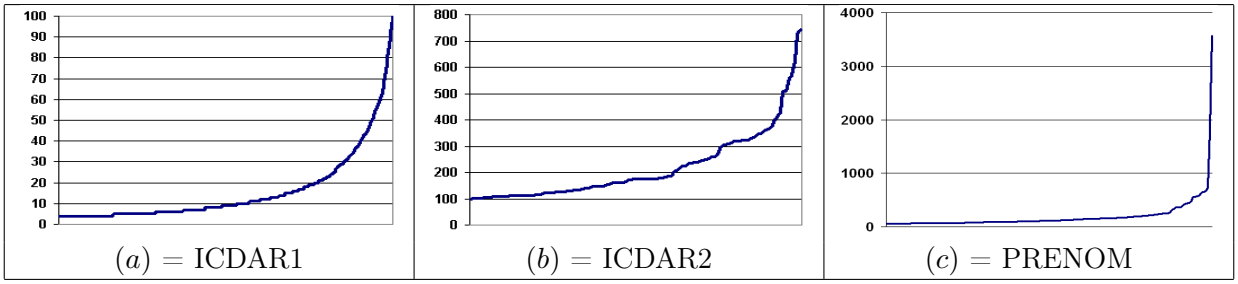


Fig. 4.27: Fréquence des mots dans les bases d'apprentissage et de test, les graphiques (a) et (b) correspondent respectivement à la première et à la seconde expérience, l'image (c) correspond à la troisième expérience.

Pour ces trois expériences, la base d'images est divisée en deux sous-bases, les bases d'apprentissage et de test de taille égale et de répartition homogène. L'expérience consiste à chercher pour les séquences d'observations de la base de tests les dix plus proches voisins de la base d'apprentissage et de classer l'image selon la classe des voisins ainsi que leur proximité.

Bien que rappelés par la table 4.11, les résultats de ces expériences ont déjà été en partie présentés au paragraphe 3.6.5 (page 67, table 3.3). Ils permettent de sélectionner le meilleur couple (segmentation en graphèmes - caractéristiques) pour une reconnaissance de mots sans avoir à comparer les performances de modèles de Markov cachés nécessitant plus d'une semaine d'apprentissage. Mis à part que les éléments à classer sont des séquences, la méthode de reconnaissance est identique à celle présentée au paragraphe 4.3.1.

expérience	jeu	taux de reconnaissance
$ICDAR1$	$Prof\,(5,5)$	14,65 %
$ICDAR1$	$Mat\,(5,5)$	20,61 %
$ICDAR2$	$Prof\,(5,5)$	45,11 %
$ICDAR2$	$Mat\,(5,5)$	52,12 %
$PRENOMS$	$Prof\,(5,5)$	30,87 %
$PRENOMS$	$Mat\,(5,5)$	40,21 %
$ICDAR2$	$Mat\,(5,5)$ + liaisons	54,75 %

Tab. 4.11: Taux de reconnaissance pour une reconnaissance de mot à l'aide de plus proches voisins. La dernière expérience reprend la base ICDAR2 mais cette fois-ci utilise une distance qui est la somme de la distance entre les deux séquences de graphèmes et de celle entre les deux séquences de liaisons.

Ces chiffres de la table 4.11 montrent tout d'abord l'importance de la taille du vocabulaire. Le taux de reconnaissance de la première expérience ICDAR1 est trois fois plus important que celui de la seconde ICDAR2 où les mots sont dix fois moins nombreux et les exemples d'apprentissages dix fois plus nombreux. Bien que ce tableau n'énumère pas les performances obtenues pour chaque jeu de caractéristiques, ces expériences ont montré que les jeux de type Mat se comportent mieux que les autres, validant en cela les résultats du paragraphe 4.3.1. La dernière expérience tend à montrer que les liaisons permettent d'accroître les performances en reconnaissance même si l'essentiel du résultat dépend de la forme des graphèmes.

L'inconvénient majeur de cette méthode est qu'il est nécessaire que la base d'apprentissage ait au moins un exemplaire du mot à reconnaître afin que le processus de reconnaissance puisse avoir une chance de l'identifier alors que les modèles de Markov cachés n'ont besoin que d'exemples de lettre. Cela ne signifie pas qu'une telle méthode de reconnaissance doive être abandonnée, la base d'apprentissage peut d'abord être complétée en générant des séquences d'observations pour les mots manquants en utilisant par exemple les méthodes décrites au paragraphe 4.7.5. Toutefois, comme un tel système est à la fois gourmand en

espace de stockage et en temps de calcul, cette extension ne paraît pas envisageable. En revanche, il serait préférable de restreindre un tel système à l'utilisation des mots fréquents tels que les mots de liaisons et ce pour deux raisons. La première est que ces mots sont disponibles en grand nombre ce qui rend un tel système performant. La seconde raison est que ces mots si fréquents sont souvent mal écrits parce que l'œil humain est habitué à les lire, ils introduisent donc du bruit lors de l'apprentissage des modèles de Markov cachés (voir paragraphe 4.5) qui eux permettent aisément de reconnaître des mots dont on ne dispose pas d'exemple dans la base d'apprentissage. Un système de reconnaissance pourrait donc allier un classifieur à base de plus proches voisins spécialisé dans la reconnaissance des mots les plus fréquents et des modèles de Markov cachés spécialisés dans la reconnaissance des mots les moins fréquents.

Cette reconnaissance pourrait également tout-à-fait être appliquée à la sélection de caractéristiques plutôt que la méthode développée au paragraphe 4.3 qui s'intéresse à la reconnaissance de caractères et non de mots. Toutefois, la distance utilisée ici est plus complexe et nécessite plus de temps de calcul, ce qui prenait deux heures pour obtenir un taux de reconnaissance caractère nécessiterait douze heures pour obtenir un taux de reconnaissance mot. La méthode présentée ici permettrait d'affiner les résultats obtenus au paragraphe 4.3.

Cette méthode de reconnaissance reconnaît les mots sans se soucier des lettres ou des syllabes. Il est d'ailleurs possible d'effectuer un parallèle avec la méthode globable d'apprentissage de la lecture qui rend difficile la lecture de nouveaux mots. Les modèles de Markov cachés, quant à eux, s'apparentent plus à la méthode syllabique puisque ceux-ci permettent de construire un modèle de reconnaissance par lettre puis de les assembler ensuite pour former un modèle de mot. Ils peuvent ainsi tout reconnaître et pallier les lacunes de la première méthode si celle-ci était utilisée en premier.

4.5 Présentation des modèles de reconnaissance

4.5.1 Modèle hybride réseau de neurones et modèles de Markov cachés

L'utilisation de modèles de reconnaissance suppose que l'image a été préalablement segmentée en graphèmes. De cette séquence de graphèmes, on tire deux séquences de vecteurs de dimension fixe, la première est la séquence d'observations $O = (O_1, \ldots, O_T)$, elle transcrit la forme des graphèmes. La seconde séquence est celle des liaisons notée $L = (L_1, \ldots, L_T)$.

Les modèles de reconnaissance (décrits dans [Augustin2001]) associent réseaux de neurones et modèles de Markov cachés et fonctionnent selon le schéma figure 4.28 afin de décrypter un mot. Le décryptage consiste ici à déterminer une probabilité mesurant la cohérence entre la séquence de graphèmes et le mot associé au modèle de reconnaissance. Une forte probabilité désigne une séquence proche de celles présentes dans la base d'apprentissage ayant servi à l'estimation des modèles.

L'ensemble d'un tel système de reconnaissance sera détaillé par la suite (aussi dans les annexes [19]). En résumé, il est composé de :

1. Un réseau de neurones classifieur, il classe chaque graphème parmi une centaine de classes, autrement dit, il détermine la classe à laquelle la forme du graphème correspond. Les liaisons ne sont pas prises en compte.
2. Un jeu de modèles de Markov cachés incluant 26 modèles, un pour chaque lettre mais il peut également intégrer des modèles associés aux signes de ponctuation, aux chiffres, aux minuscules, aux majuscules. Cette liste dépendant des hypothèses du problème à résoudre.

Les performances des modèles hybrides ainsi construits sont comparées pour une même séquence. Si deux mots m_1 et m_2 ont pour modèles de Markov cachés M_1 et M_2, le mot m_1 est plus probable que le mot m_2

19. Annexes : voir paragraphe E, page 245

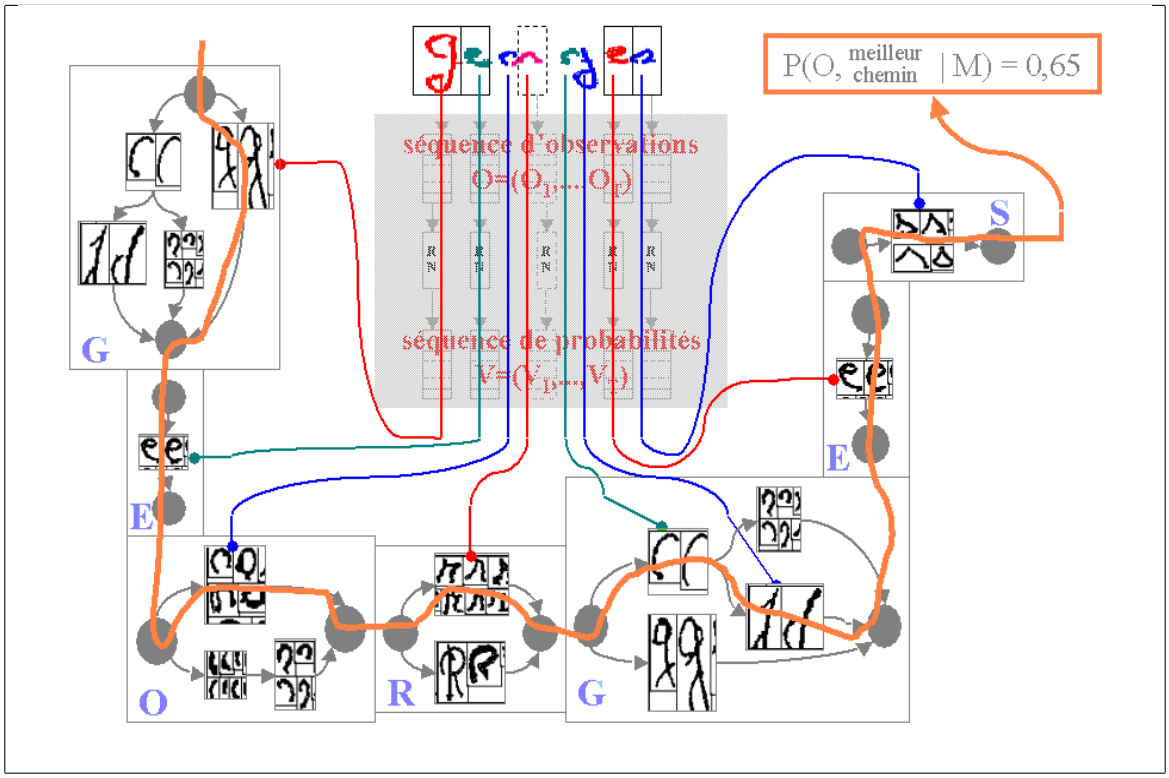


Fig. 4.28: Décryptage d'un mot à l'aide d'un modèle hybride réseau de neurones et modèle de Markov caché. Le réseau de neurones permet de reconnaître en classant chaque graphème parmi la centaine de classes disponibles. Le modèle de Markov caché associé au mot "Georges" résulte de la juxtaposition des modèles associés aux lettres qui le composent. Chacun d'entre eux est illustré par les séquences de classes de graphèmes les plus courantes qui permettent d'écrire une lettre. La ligne parcourant successivement les modèles des lettres du mot "Georges" correspond à la meilleure association entre la séquence d'observations extraites de l'image et la juxtaposition de celles apprises par chacune des lettres.

si $\mathbb{P}(O \mid M_1) > \mathbb{P}(O \mid M_2)$. L'expression de cette probabilité ainsi que les modèles M_1 et M_2 sont détaillés dans les paragraphes qui suivent.

4.5.2 Liaisons

Les modèles présentés au paragraphe précédent s'appuient sur une séquence d'observations $(O_1, \ldots, O_T)$ décrivant la forme des graphèmes. Comme le montre la figure 4.29, cette information n'est pas toujours suffisante pour déterminer une séquence de lettres. La première idée consiste à ajouter un vecteur de caractéristiques décrivant cette liaison à celui décrivant la forme. La dimension des vecteurs de la séquence $(O_1, \ldots, O_T)$ en est augmentée ainsi que le nombre de classes d'observations nécessaires pour classer ces nouvelles observations. Cette seconde séquence est notée $(L_1, \ldots, L_T)$. Bien qu'il n'y ait que $T-1$ liaisons pour T graphèmes, afin de conserver les mêmes dimensions, la liaison L_1 désigne le début du mot et la valeur qui lui est affectée décrit une liaison entre graphèmes infiniment éloignés. Les autres liaisons L_t décrivent la liaison entre les graphèmes O_{t-1} et O_t.

Si on note C_G le nombre de classes de graphèmes et C_L le nombre de classes de liaisons, le nombre de classes nécessaires pour classer les observations est susceptible d'atteindre la borne $C_G \times C_L$. La première idée consiste à regrouper ces deux séquences en une seule $\left(\binom{O_1}{L_1}, \ldots, \binom{O_T}{L_T}\right)$ puis d'utiliser les mêmes types modèles avec ces nouvelles données. Quelques expériences ont été menées en ce sens et les résultats obtenus n'ont pas été meilleurs que ceux obtenus avec la séquence $(O_1, \ldots, O_T)$. Par conséquent, cette

Fig. 4.29: Information contenue dans les liaisons : sur cette segmentation graphème, la lettre "G" est découpée en deux morceaux, le premier ressemble à un "C", le second à un "J". Si aucune information relative à la liaison entre ces deux graphèmes n'est prise en compte, la séquence "CJ" est tout aussi probable que la séquence "G".

voie a été abandonnée pour construire des modèles prenant en compte deux séquences d'observations, la première décrivant les graphèmes, la seconde décrivant les liaisons. Ces modèles sont inspirés des Input-Output Hidden Markov Models (IOHMM) développés dans [Bengio1996]. Les vecteurs inclus dans les séquences doivent d'abord être probabilisés avant d'être utilisés par des modèles de Markov cachés. Parmi les options les plus courantes, on distingue la modélisation de la densité des observations par un mélange de lois gaussiennes ou la classification des observations de manière à obtenir les probabilités suivantes :

– séquence des probabilités des classes d'observations : $\left(C_1^O, \ldots, C_T^O\right)$ et

$$C_i^O = \left(C_{i,1}^O, \ldots, C_{i,N_O}^O\right)' \in [0,1]^{N_O} \text{ et } \forall k,\ C_{i,k}^O = \mathbb{P}\left(O_i \in \textit{classe } k \mid O_i\right)$$

– séquence des probabilités des classes de liaisons : $\left(C_1^L, \ldots, C_T^L\right)$

$$C_i^L = \left(C_{i,1}^L, \ldots, C_{i,N_L}^L\right)' \in [0,1]^{N_L} \text{ et } \forall k,\ C_{i,k}^L = \mathbb{P}\left(L_i \in \textit{classe } k \mid L_i\right)$$

Les modèles développés par [Augustin2001] utilisent les réseaux de neurones, comme ces travaux s'incrivent dans la continuité de ceux de [Augustin2001], c'est celle-ci qui a été choisie. Ces probabilités sont le résultat de réseaux de neurones classifieurs [20]. Les modèles IOHMM présentés ici seront toujours hybrides alliant réseau de neurones pour reconnaître les formes ou les liaisons et modèles de Markov cachés pour modéliser les séquences.

4.5.3 Modèles de lettres

A chaque mot est associé un modèle conçu comme étant la juxtaposition de modèles de lettres que ce paragraphe a pour but d'introduire. Ces modèles de lettres constituent la brique élémentaire du système de reconnaissance. Soit q_t l'état d'une chaîne de Markov à l'instant t et C_t^L la classe de la L_t à l'instant t, voici l'hypothèse supplémentaire du modèle IOHMM par rapport au modèle HMM [21] :

$$\begin{aligned}
\mathbb{P}\left(q_t \mid q_{t-1}, \overline{L_t}\right) &= \sum_{i=1}^{N_l} \mathbb{P}\left(q_t, C_t^L = i \mid q_{t-1}, L_t\right) \\
&= \sum_{i=1}^{N_l} \mathbb{P}\left(q_t \mid C_t^L = i, q_{t-1}\right)\ \mathbb{P}\left(C_t^L = i \mid q_{t-1}, L_t\right) \\
&= \sum_{i=1}^{N_l} \mathbb{P}\left(q_t \mid C_t^L = i, q_{t-1}\right)\ \mathbb{P}\left(C_t^L = i \mid L_t\right)
\end{aligned}$$

20. Annexes : voir paragraphe C.1.5, page 196
21. Annexes : voir paragraphe E.2, page 247

notation	signification	modèle attaché
$c_{i,c}$	$\mathbb{P}\left(C_t^O = c \mid q_t = i\right)$	MMC
$\gamma_c\left(O_t\right)$	$\mathbb{P}\left(C_t^O = c \mid O_t\right)$	RN^O
$a_{i,j,d}$	$\mathbb{P}\left(q_t = j \mid q_{t-1}, C_t^L = d\right)$	MMC
$\pi_{j,d}$	$\mathbb{P}\left(q_1 = j \mid C_1^L = d\right)$	MMC
$\delta_d\left(L_t\right)$	$\mathbb{P}\left(C_t^L = d \mid L_t\right)$	RN^L
θ_j	$\mathbb{P}\left(fin \mid q_T = j\right)$	MMC
$f\left(L_t\right)$	densité des liaisons	–
$g\left(O_t\right)$	densité des observations	–
p_c	$\mathbb{P}\left(C_t^0 = c\right)$	–

Tab. 4.12: Notations utilisées pour désigner les paramètres des modèles IOHMM hybrides.

$$\begin{array}{|rcl|rcl|}
\hline
\forall i,\ \sum\limits_c c_{i,c} & = & 1 & \sum\limits_c \gamma_c\left(O_t\right) & = & 1 \\
\forall i,\ \sum\limits_j \left[\theta_j + \sum\limits_d a_{i,j,d}\right] & = & 1 & \forall i,\ \sum\limits_d \pi_{i,d} & = & 1 \\
\sum\limits_d \delta_d\left(O_t\right) & = & 1 & \sum\limits_c p_c & = & 1 \\
\int f\left(L\right) dL & = & 1 & \int g\left(O\right) dO & = & 1 \\
\hline
\end{array} \tag{4.39}$$

Tab. 4.13: Contraintes vérifiées par les coefficients donnés par la table 4.12.

La table 4.12 regroupe la liste des paramètres utilisés pour le calcul des probabilités des séquences ainsi que pour l'apprentissage. D'après la définition des IOHMM, ces paramètres doivent vérifier les contraintes énoncées dans la table 4.13. Avec ces notations, la probabilité d'un chemin ou séquence d'états du IOHMM devient :

$$\mathbb{P}\left(O_1, \ldots, O_T,\ L_1, \ldots, L_T,\ q_1, \ldots, q_T\right) = \pi_{q_1}\left(L_1\right)\ b_{q_1}\left(O_1\right)\ \left[\prod_{t=2}^{T} a_{q_{t-1},q_t}\left(L_t\right)\ b_{q_t}\left(O_t\right)\right] \theta_{q_T}$$

$$\text{avec } a_{q_{t-1},q_t}\left(L_t\right) = \sum_{c=1}^{N_C} \mathbb{P}\left(q_t \mid q_{t-1}, L_t\right)\ \underbrace{f\left(L_t\right)}_{\substack{\text{densité} \\ \text{des liaisons}}}$$

Afin de simplifier les expressions résultant du calcul des probabilités d'émission, les expressions intermédiaires suivantes sont calculées à chaque itération t :

$$b_i\left(O_t\right) = g\left(O_t\right) \sum_{c=1}^{N_O} \frac{\gamma_c\left(O_t\right)\ c_{i,c}}{p_c} \tag{4.40}$$

$$a_{i,j}\left(L_t\right) = f\left(L_t\right) \sum_{c=1}^{N_L} a_{i,j,c}\ \delta_c\left(L_t\right) \tag{4.41}$$

$$\pi_i\left(L_1\right) = f\left(L_1\right) \sum_{c=1}^{N_L} \pi_{i,c}\ \delta_c\left(L_1\right) \tag{4.42}$$

L'objectif est tout d'abord de calculer la probabilité d'émission des deux séquences d'observations et de liaisons notée :

$$\mathbb{P}\left(O_1,\ldots,O_T,\ L_1,\ldots,L_T\right) = \sum_{(q_1,\ldots,q_T)} \mathbb{P}\left(O_1,\ldots,O_T,\ L_1,\ldots,L_T,\ q_1,\ldots,q_T\right) \tag{4.43}$$

Comme pour le calcul de la probabilité d'émission d'une séquence d'émission dans le cas d'un modèle de Markov caché [22], on construit les suites $(\alpha_{t,i})$ et $(\beta_{t,i})$ définies par :

$$\alpha_{t,i} = \mathbb{P}\left(q_t = i,\ O_1,\ldots,O_t,\ L_1,\ldots,L_t\right) \tag{4.44}$$
$$\beta_{t,i} = \mathbb{P}\left(O_{t+1},\ldots,O_T,\ L_{t+1},\ldots,L_T \mid q_t = i\right) \tag{4.45}$$
$$\beta'_{t,i} = \beta_{t,i} b_i\left(O_t\right) \tag{4.46}$$

La dernière suite est plus souvent utilisée car elle est plus commode dans le calcul des probabilités. De manière analogue à l'algorithme E.2.9, la suite $(\alpha_{t,i})$ est calculée de la manière suivante :

$$\forall i,\ \alpha_{1,i} = \pi_i\left(L_t\right)\ b_i\left(O_1\right) \tag{4.47}$$
$$\forall j,\ \forall t,\ \alpha_{t,j} = \sum_{i=1}^{Q} \alpha_{t-1,i}\ a_{i,j}\left(L_t\right)\ b_j\left(O_t\right) \tag{4.48}$$
$$\mathbb{P}\left(O_1,\ldots,O_T,\ L_1,\ldots,L_T\right) = \sum_{i=1}^{Q} \alpha_{T,i}\ \theta_i \tag{4.49}$$

De manière analogue à l'algorithme E.2.10, la suite $\left(\beta'_{t,i}\right)$ est calculée de la manière suivante :

$$\forall i,\ \beta'_{T,i} = \theta_i\ b_i\left(O_T\right) \tag{4.50}$$
$$\forall i,\ \forall t,\ \beta'_{t,i} = b_i\left(O_t\right) \sum_{j=1}^{Q} a_{i,j}\left(L_{t+1}\right)\ \beta'_{t+1,j} \tag{4.51}$$
$$\mathbb{P}\left(O_1,\ldots,O_T,\ L_1,\ldots,L_T\right) = \sum_{i=1}^{Q} \pi_i\left(L_1\right)\ \beta'_{i,1} \tag{4.52}$$

Les formules précédentes permettent d'utiliser un modèle IOHMM, il ne reste plus qu'à apprendre les paramètres qui le composent. Cet apprentissage s'effectue à partir d'une base de données contenant pour chaque image d'indice k, une séquence d'observations $\left(O_1^k,\ldots,O_{T_k}^k\right)$, une séquence de liaisons $\left(L_1^k,\ldots,L_{T_k}^k\right)$ et une annotation A_k correspondant au contenu de l'image [23] Les notations intermédiaires suivantes sont utilisées :

22. Annexes : voir paragraphe E.2.3, page 250

23. Cette information est souvent le mot écrit dans cette image, puisqu'à chaque lettre est associé un modèle, l'annotation permet de déterminer quels modèles doivent inclure dans leur apprentissage les séquences d'observations et de liaisons extraites de l'image. Etant donné qu'un modèle de mot est la juxtaposition de plusieurs modèles de lettres, les formules de la table 4.14 seront utilisées pour chaque modèle de lettre et chaque sous-séquence extraites des séquences d'observations et de liaisons pondérées par leur vraisemblance (voir paragraphe 4.6.3).

$$\overline{\theta_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k}\ \alpha_{T_k,i}^k\ \beta_{T_k,i}^k}{\sum_{k=1}^{K} \frac{1}{P_k}\ \left[\sum_{t=1}^{T_k} \alpha_{t,i}^k\ \beta_{t,i}^k\right]} \tag{4.56}$$

$$\overline{\pi_{i,d}} = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{P_k}\ \alpha_{1,i}^k\ \delta_d\left(L_1\right)\ \beta_{1,i}^k \tag{4.57}$$

$$\overline{a_{i,j,d}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k}\ \left[\sum_{t=1}^{T_k-1} \alpha_{t,i}^k\ a_{i,j,d}\ \delta_d\left(L_t\right)\ \beta_{t+1,j}^{\prime k}\right]}{\sum_{k=1}^{K} \frac{1}{P_k}\ \left[\sum_{t=1}^{T_k} \alpha_{t,i}^k\ \beta_{t,i}^k\right]} \tag{4.58}$$

$$\overline{c_{i,c}} = \frac{\sum_{k=1}^{K} \frac{1}{K} \sum_{t=1}^{T_k} \frac{\alpha_{t,i}^k\ c_{i,c}\ \gamma_c\left(O_t\right)\ \beta_{t,i}^k}{\sum_{u=1}^{N_O} c_{i,u}\ \gamma_u\left(O_t\right)}}{\sum_{k=1}^{K} \frac{1}{P_k}\ \left[\sum_{t=1}^{T_k} \alpha_{t,i}^k\ \beta_{t,i}^k\right]} \tag{4.59}$$

Tab. 4.14: Formules de mise à jour des paramètres afférents à la partie chaîne de Markov cachée d'un modèle IOHMM.

$$P_k = \mathbb{P}\left(O_1^k, \ldots, O_{T_k}^k, L_1^k, \ldots, L_{T_k}^k\right) \tag{4.53}$$

$$\alpha_{t,i}^k = \mathbb{P}\left(q_t = i,\ O_1^k, \ldots, O_t^k,\ L_1^k, \ldots, L_t^k\right) \tag{4.54}$$

$$\beta_{t,i}^k = \mathbb{P}\left(O_{t+1}^k, \ldots, O_T^k,\ L_{t+1}^k, \ldots, L_T^k \mid q_t = i\right) \tag{4.55}$$

Les formules de mise à jour des paramètres d'un IOHMM sont décrites dans la table 4.14, celles-ci s'inspirent de celle d'un modèle de Markov caché classique [24] et se démontrent de la même manière. Les réseaux de neurones classifieurs sont eux-aussi appris en construisant des bases d'apprentissages formées des vecteurs suivants pour les observations [25] (X, Y), $Y = RN^O(X)$:

$$\left|\begin{array}{rcl} X &=& O_t^k \\ Y &=& \left(\frac{1}{P_k} \sum_i \frac{\alpha_{t,i}^k\ c_{i,c}\ \gamma_c\left(O_t^k\right)\ \beta_{t,i}^k}{\sum_{u=1}^{N_O} c_{i,u}\ \gamma_u\left(O_t^k\right)}\right)_{1 \leqslant c \leqslant N_O} \end{array}\right. \tag{4.60}$$

24. Annexes : voir paragraphe E.4.1, page 263
25. Annexes : voir paragraphe E.5, page 281

Si on note Y_{tc}^k la $c^{\text{ème}}$ sortie désirée du réseau de neurones RN^O pour le vecteur d'observations O_t^k, celle donnée par la formule ci-dessus ne maximise pas la vraisemblance mais il est possible de déterminer cette valeur en appliquant l'algorithme EM de sortie que :

$$\overline{Y_{tc}^k} = \frac{1}{P_k} \left(\sum_{q_t} \frac{\alpha_{q_t}^k(t)\, Y_{tc}^k\, c_{q_t,c}\, \beta_{q_t}^k(t)}{\sum_{u=1}^{C} Y_{tu}^k\, c_{q_t,u}} \right) \tag{4.61}$$

Le vecteur Y de l'expression (4.60) est dans ce cas composé du vecteur $\left(\overline{Y_{t1}^k}, \ldots, \overline{Y_{tN^O}^k}\right)$ obtenu après convergence de la formule (4.61). La valeur initiale de Y_{tc}^k est égale à $\gamma_c\left(O_t^k\right)$. Les suites $\alpha(.)$ et $\beta(.)$ sont bien sûr recalculées entre deux mises à jour. On obtient un résultat analogue en ce qui concerne les liaisons, (X, Y), $Y = RN^L(X)$:

$$\left| \begin{array}{lcl} X & = & L_t^k \\ Y & = & \left(\frac{1}{P_k} \sum\limits_{i,j} \alpha_{t,i}^k\, a_{i,j,d}\, \delta_d\left(L_t^k\right)\, \beta_{t+1,j}'^k \right)_{1 \leqslant d \leqslant N_L} \end{array} \right. \tag{4.62}$$

La remarque précédente concernant les sorties désirées pour le réseau de neurones associées aux observations est valable pour les liaisons, le vecteur Y de l'expression (4.62) est composé du vecteur $\left(\overline{Y_{t1}^k}, \ldots, \overline{Y_{tN^L}^k}\right)$ obtenu après convergence de la formule (4.63). La valeur initiale de Y_{td}^k est égale à $\delta_d\left(L_t^k\right)$. Les suites $\alpha(.)$ et $\beta(.)$ sont bien sûr recalculées entre deux mises à jour.

$$\overline{Y_{td}^k} = \frac{1}{P_k} \sum_{i,j} \alpha_{t,i}^k\, a_{i,j,d}\, Y_{td}^k\, \beta_{t+1,j}'^k \tag{4.63}$$

Une dernière remarque concerne les sorties désirées obtenus pour l'apprentissage des réseaux de neurones de classifications des graphèmes et des liaisons. Ces sorties sont souvent soient nulles soient égales à un et ces valeurs ne facilitent pas l'apprentissage de ces réseaux comme le suggère la remarque C.5.6 [26] qui propose une modification de ces sorties permettant de faciliter leur apprentissage.

4.5.4 Topologie des modèles

Les formules d'apprentissage qui s'appliquent aux IOHMM présentés au paragraphe 4.5.2 nécessitent que la structure des modèles soit fixée au préalable. Il faut éviter que celle-ci contienne trop de coefficients auquel cas le calcul des probabilités est coûteux et les modèles risquent de faire du sur-apprentissage, ni trop peu de coefficients afin que les modèles puissent véritablement identifier la lettre à laquelle ils sont affectés.

Dans un premier temps, la structure est fixée arbitrairement en tenant compte de la complexité de chaque lettre, celle-ci est fonction du nombre moyen de graphèmes par lettre. Il est estimé manuellement à partir d'une centaine de documents et donné par la table 4.15. L'article [Günter2003] propose quant à lui de construire des modèles de Markov simplifiés afin d'estimer statistiquement la longueur moyenne des séquences associées à chaque modèle de lettre.

26. Annexes : voir paragraphe C.5.6, page 223

a	2	f	2	k	2	p	2	u	2	z	2
b	2	g	2	l	1	q	2	v	2		
c	2	h	2	m	3	r	2	w	3		
d	2	i	1	n	2	s	2	x	2		
e	2	j	2	o	2	t	2	y	2		

Tab. 4.15: Nombre moyen de graphèmes par lettre : ce nombre est estimé sur une centaine de documents par un opérateur humain, les moyennes sont arrondies au premier entier supérieur.. Une fois les modèles appris, il pourra être affiné grâce à un algorithme de Viterbi permettant de trouver la meilleure association entre les graphèmes et les modèles de lettres. Les minuscules et majuscules ne sont pas dissociées.

Le nombre d'états e_k choisi pour une lettre k est fonction de g_k le nombre moyen de graphème donné par la table 4.15. Après plusieurs expériences, la règle suivante semble être pertinente :

$$e_k = x\,(g_k + 1) \text{ avec } x \in \mathbb{N} \tag{4.64}$$

La matrice des transitions est triangulaire strictement supérieure de manière à éviter les cycles. Les coefficients non nuls relatifs aux classes d'observations sont limités à trois pour un état ($c_{i,c}$), ce nombre permet de retrouver un nombre de coefficients légèrement supérieur à ceux observés dans les modèles développés par [Augustin2001]. Par conséquent, on pose $x = 3$, le nombre de coefficients d'un modèle M_k associé à la lettre k correspond à :

$$\underbrace{e_k^2 C_L}_{\text{probabilités de transitions}} + \underbrace{e_k C_L}_{\text{probabilités d'entrée}} + \underbrace{e_k}_{\text{probabilités de sortie}} + \underbrace{3e_k}_{\text{probabilités d'émission}}$$

et C_L est le nombre de classes liaison.

Il est préférable de surestimer le nombre optimal de coefficients de façon à pouvoir faire décroître ce nombre en supprimant les coefficients inutiles (voir paragraphe 4.9).

Le choix d'une topologie est difficile. L'article [Bengio1995] étudie quatre différentes topologies :

1. Le modèle périodique, chaque état boucle est connecté à lui-même et à l'état suivant, formant une chaîne d'état.
2. Le modèle gauche-droite, semblable à un graphe.
3. Le modèle gauche-droite triangulaire, celui choisi dans ce paragraphe, pour lequel la matrice de transition est triangulaire.
4. Le modèle entièrement connecté.

Cet article mesure la complexité la difficulté de l'estimation de chacun de ces modèles par l'intermédiaire du coefficient d'ergodicité de Dobrushin (voir [Senta1986]) :

$$\tau(A) = \frac{1}{2}\sup_{i,j} |a_{ik} - a_{jk}| \text{ où } A \text{ est la matrice de transition} \tag{4.65}$$

Ce coefficient d'ergodicité converge vers 0 lors de l'apprentissage, il converge d'autant plus vite que la topologie des modèles est complexe. Plus il converge vite, plus l'apprentissage est difficile. L'article [Bengio1995] étudie cette convergence pour les quatre types de topologie présentés ci-dessus. L'article

[Abou-Mustafa2004] reprend ces résultats et conclut par le fait que les topologies les plus simples sont plus à même de généraliser et que les résultats de reconnaissance sont meilleurs lorsque les probabilités sont déterministes, c'est-à-dire proche de 0 ou 1.

4.6 Reconnaissance avec dictionnaire

4.6.1 Principe

On considère une séquence d'observations obtenue à partir d'une image ségmentée en graphèmes (figure 4.30), la reconnaissance avec dictionnaire consiste à trouver le mot dans cette liste qui correspond à l'image.

Fig. 4.30: Un mot segmenté en graphèmes

La définition qui suit introduit les notations qui seront couramment utilisées par la suite.

Définition 4.6.1 : reconnaissance avec dictionnaire
Soit $O = (O_1, \ldots, O_T) \in \mathcal{O}$ une séquence d'observations avec $\mathcal{O}$ l'ensemble des séquences d'observations, cette séquence décrit une image du mot m^*. Soit un dictionnaire $D = (m_1, \ldots, m_N)$ ou liste finie de mots, la reconnaissance avec le dictionnaire D telle qu'elle est présentée dans ce document consiste à trouver une fonction $f_D : (\mathcal{O}, D) \rightarrow \mathbb{R}$ telle que :

$$\arg\max_{m \in D} f_D(O, m) = m^*$$

Remarque 4.6.2: rejet
Cette définition sous-entend que la solution m^* appartient au dictionnaire et qu'il n'est pas possible de rejeter un mot n'en faisant pas partie. Il est cependant possible d'accepter ou de refuser la solution selon la valeur de $f_D(O, m^*)$.

On impose également au dictionnaire d'être dynamique (définition 4.6.3). Même si la définition du problème ne s'en trouve pas changée, les choix de modélisation doivent tenir compte de cette contrainte supplémentaire.

Définition 4.6.3 : dictionnaire dynamique
Un dictionnaire D est dit dynamique si l'ajout ou la suppression de mot n'implique pas une nouvelle recherche de la fonction $f_{D'}$ où D' est un dictionnaire différent de D. L'estimation de la fonction f est par conséquent indépendante du dictionnaire D.

La solution préconisée est la construction pour un mot m d'un modèle probabiliste de mot M_m associé à ce mot. On note $\mathbb{P}(M \mid O)$ la probabilité du modèle sachant la séquence d'observations.

$$f(O, m) = \mathbb{P}(M_m \mid O) = \mathbb{P}(O \mid M_m) \frac{\mathbb{P}(M_m)}{\mathbb{P}(O)} \tag{4.66}$$

On suppose que $\mathbb{P}(M)$ est constante quel que soit le mot m et dans ce cas :

$$\arg\max_{m \in D} f_D(O, m) = \arg\max_{m \in D} \mathbb{P}(O \mid M_m) \tag{4.67}$$

La segmentation en graphèmes n'a pas encore été justifiée tout comme l'utilisation d'une séquence d'observations. Il est tout-à-fait possible de construire un modèle de mot prenant en compte l'image dans son ensemble et non des morceaux. Mais un simple réseau de neurones ou tout autre classifieur chargé de reconnaître le mot "CHARLES" par exemple devra être estimé sur une base d'apprentissage incluant des images de ce mot. Il est alors impossible de reconnaître ce mot s'il n'est pas présent dans les bases d'apprentissage (voir paragraphe 4.4) ce qui se produit lorsque le dictionnaire est dynamique comme le montre la table 4.16 qui décrit les occurrences des prénoms dans les bases d'apprentissage et de test pour un problème de reconnaissance de prénoms manuscrits.

prénoms	occurrences dans la base d'apprentissage	occurrences dans la base de test
JEAN	1796	615
JEANE	0	1
JEANETTE	0	1
JEANINE	34	9
JEANNE	636	186
JEANNETTE	15	8
JEANNIE	2	0
JEANNINE	99	37
JEHAN	2	0
JEMMY	1	0
JENNY	20	2
JERD	1	0
JEREMIE	1	1
JEROME	15	5

Tab. 4.16: Occurrences de prénoms dans les bases d'apprentissage et de test. Certains mots sont présents dans la base de test et pas dans la base d'apprentissage. Cela exclut leur apprentissage par un modèle de mot n'utilisant aucune segmentation d'image comme celle du paragraphe 4.4.

L'option retenue, la plus simple dans sa conception mais pas forcément dans sa mise en œuvre, est la segmentation d'un mot en graphèmes ou imagettes respectant les contraintes suivantes :

1. Chaque graphème est inclus dans le dessin d'une lettre, c'est donc une lettre ou une partie de lettre.
2. Les graphèmes doivent être ordonnés selon un ordre respectant l'ordre des lettres dans le mot.

Par conséquent, pour l'image du mot m dont les lettres sont $m = (l_1, \ldots, l_N)$, on dispose de la séquence de graphèmes $G = (G_1, \ldots, G_T)$ extraite de l'image et G_t représente une partie de la lettre l_{a_t} où a_t est l'indice de la lettre qui contient le graphème d'indice t. Ces trois séquences vérifient :

1. $N \leqslant T$
2. $a_1 = 1$ et $a_T = N$
3. si $t < T$, alors $a_t \leqslant a_{t+1} \leqslant \min\{a_t + 1, N\}$

Pour le processus de reconnaissance, il aurait été plus facile de segmenter l'image d'un mot en lettres mais ce traitement d'image est difficile à réaliser, c'est pourquoi cette segmentation plus souple est choisie (voir [Augustin2001]) et même celle-ci n'est pas toujours facile à respecter (voir paragraphe 4.7). Les graphèmes obtenus par segmentation doivent à la fois être suffisamment gros pour être reconnus et ordonnés, suffisamment petits pour ne pas représenter plus d'une lettre.

Les contraintes énoncées ci-dessus sont également valables pour la séquence $O = (O_1, \ldots, O_T)$ qui est la transcription de la séquence $G = (G_1, \ldots, G_T)$ dans un format utilisable par les modèles de Markov cachés et les réseaux de neurones.

La séquence $O = (O_1, \ldots, O_T)$ est donc une succession de lettres ou morceaux de lettres respectant l'ordre de lecture. Ce découpage autorise la construction de modèles de reconnaissance de lettres à partir desquels seront formés les modèles de mots. Par conséquent, la fonction $f(O, m)$ sera uniquement dépendante de vingt-six modèles de lettres et de la séquence d'observations. Le paragraphe suivant montre comment est construit un modèle M_m associé au mot m.

4.6.2 Construction de modèles de mot

Les modèles utilisés sont des modèles de Markov cachés hybrides associés à un réseau de neurones [27]. La figure 4.28 illustre quelques modèles de lettres [28]. Si le mot $m = (l_1, \ldots, l_N)$, les modèles de lettres $H = (H_1, \ldots, H_N)$ seront assemblés de manière à former le modèle M_m comme montré par la figure 4.31. Cet assemblage est simplement une juxtaposition comme le montre la figure 4.28.

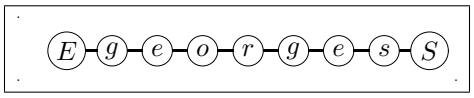


Fig. 4.31: Modèle de mot pour "georges" construit à partir des modèles de lettres assoicés aux lettres "g", "e", "o", "r", "s". Les symboles "E", "S" signifient l'entrée et la sortie du modèle.

Maintenant que les modèles de mot sont définis, pour un mot $m = (l_1, \ldots, l_N)$, il s'agit de calculer la probabilité $\mathbb{P}(O \mid M_m)$ des deux séquences $(O, L) = (O_1, \ldots, O_T, L_1, \ldots, L_T)$. Le modèle M_m est la juxtaposition des modèles $M_m = (H_{l_1}, \ldots, H_{l_N})$. Pour simplifier, cette séquence sera notée $M_m = (H_1, \ldots, H_N)$.

Théorème 4.6.4 : probabilité d'une séquence avec un modèle de mot
Soit un modèle de mot M_m composé de modèles de lettre $M_m = (H_1, \ldots, H_N)$ et les séquences d'observations et de liaisons $(O, L) = (O_1, \ldots, O_T, L_1, \ldots, L_T)$. On définit la suite $\left(\alpha_{t,t'}^{H_i}\right)_{t,t',i}$ par :

$$\forall (t, t') \in \{1, \cdots, T\}^2, \forall i \in \{1, \cdots, N\} \ \alpha_{t,t'}^{H_i} = \begin{cases} \mathbb{P}(O_t, \ldots, O_{t'}, L_t, \ldots, L_{t'} \mid H_i) \text{ si } t \leqslant t' \\ 0 \text{ sinon} \end{cases} \tag{4.68}$$

Le calcul de cette suite fait appel à l'équation (E.7). On définit également la suite $\left(\gamma_t^{H_i}\right)_{t,i}$ par :

$$\begin{aligned} \forall i \in \{1, \cdots, N\}, \quad \gamma_0^{H_i} &= 1 \\ \forall i \in \{1, \cdots, N\}, \forall t \in \{1, \cdots, T\}, \quad \gamma_t^{H_i} &= \sum_{k=1}^{t-1} \gamma_k^{H_{i-1}} \alpha_{k+1,t}^{H_i} \end{aligned} \tag{4.69}$$

Alors :

$$\mathbb{P}(O \mid M_m) = \gamma_T^{H_N} \tag{4.70}$$

27. Annexes : voir paragraphe E.5.1, page 281
28. Annexes : voir paragraphe A, page 154

Démonstration (théorème 4.6.4) :
La démonstration de ce théorème est similaire à celle de l'équation (E.7). **(4.6.4)** □

Ce théorème permet de définir l'algorithme suivant :

Algorithme 4.6.5 : probabilité d'une séquence avec un modèle de mot
Etape A : initialisation 1
Calcul de la suite $\left(\alpha_{t,t'}^{H_i}\right)_{t,t',i}$ grâce à l'algorithme E.2.3.

Etape B : initialisation 2
pour $i = 1$ **à** N **faire**
$\gamma_0^{L_i} \longleftarrow 1$
fin pour

Etape C : itérations
pour $t = 1$ **à** T **faire**
pour $i = 1$ **à** N **faire**
$\gamma_t^{H_i} \longleftarrow 0$
pour $k = 1$ **à** $t-1$ **faire**
$\gamma_t^{H_i} \longleftarrow \gamma_t^{H_i} + \gamma_k^{H_{i-1}}\, \alpha_{k+1,t}^{H_i}$
fin pour
fin pour
fin pour

La probabilité cherchée est $\gamma_T^{H_N}$.

4.6.3 Apprentissage des modèles de lettre

L'apprentissage des modèles de lettre s'appuie sur celui des modèles de Markov cachés. Toutefois, comme ceux-ci sont utilisés dans des modèles de mot, les formules données tableau E.2 ne s'appliquent pas telles quelles.

Pour simplifier, supposons que le dictionnaire se réduit à deux mots "CHARLES" et "CAROLE". Les modèles respectifs à ces deux mots sont notés M_{CH} et M_{CA} de paramètres Θ_{CH} et Θ_{CA} qui regroupent l'ensemble des paramètres de chacun des modèles de lettres qui les composent. L'estimation de Θ_{CH} et Θ_{CA} est toujours un problème d'optimisation sur une base d'apprentissage contenant plusieurs exemples de mots "CHARLES" et "CAROLE", les séquences correspondantes sont notées :

$$O^{CH} = \left(O_1^{CH}, \ldots, O_{N_{CH}}^{CH}\right) \text{ et } O^{CA} = \left(O_1^{CA}, \ldots, O_{N_{CA}}^{CA}\right)$$

La vraisemblance des paramètres $(\Theta_{CH}, \Theta_{CA})$ est :

$$\begin{aligned} L\left(\Theta_{CH}, \Theta_{CA}, O^{CH}, O^{CA}\right) &= \prod_{n=1}^{N_{CH}} \mathbb{P}\left(O_n^{CH} \mid M_{CH}\right) \prod_{n=1}^{K_{CA}} \mathbb{P}\left(O_n^{CA} \mid M_{CA}\right) \\ &= \prod_{n=1}^{N_{CH}} \mathbb{P}\left(O_n^{CH} \mid \Theta_{CH}\right) \prod_{n=1}^{K_{CA}} \mathbb{P}\left(O_n^{CA} \mid \Theta_{CA}\right) \end{aligned}$$

Finalement, la log-vraisemblance est :

$$\ln L\left(\Theta_{CH},\Theta_{CA},O^{CH},O^{CA}\right)=\sum_{n=1}^{N_{CH}}\ln\mathbb{P}\left(O_n^{CH}\mid\Theta_{CH}\right)+\sum_{n=1}^{N_{CA}}\ln\mathbb{P}\left(O_n^{CA}\mid\Theta_{CA}\right) \tag{4.71}$$

Si les modèles M_{CH} et M_{CA} n'ont aucun paramètre commun, l'optimisation de (4.71) est équivalente à deux optimisations séparées de Θ_{CH} et Θ_{CA}. Les mots "CHARLES" et "CAROLE" ont cependant les lettres "C", "A", "L", "E", "R" en commun. L'apprentissage de tels modèles est résolu dans les paragraphes qui suivent, les formules de réestimation des paramètres des modèles de lettres ne seront pas démontrées mais les raisonnements développés dans les paragraphes E.4.2 à E.4.5 sont encore valables.

Toujours pour deux séquences d'observations et de liaisons $(O,L)=(O_1,\ldots,O_T,L_1,\ldots,L_T)$, un modèle $M_m=(H_1,\ldots,H_N)$, on note q un état d'un modèle de lettre, la notation $q\in H_i$ désigne un état appartenant au modèle H_i, q_t désigne l'état à l'instant t. On définit la suite :

$$\forall t\in\{1,\cdots,T\},\,\forall i\in\{1,\cdots,N\},\,\forall q\in H_i,\;\;\alpha_t^{H_i}(q)=\mathbb{P}\left(O_1,\ldots,O_t,L_1,\ldots,L_t,\,q_t=q\mid M_m\right) \tag{4.72}$$

Cette suite est semblable à celle définie (E.4) mais avec un indice de plus pour le modèle de lettre concerné. De même, la suite définie en (E.8) est déclinée pour un modèle de mot :

$$\forall t\in\{1,\cdots,T\},\,\forall i\in\{1,\cdots,N\},\,\forall q\in H_i,\;\;\beta_t^{H_i}(q)=\mathbb{P}\left(O_{t+1},\ldots,O_T,L_{t+1},\ldots,L_T\mid q_t=q,\,M_m\right) \tag{4.73}$$

La suite (4.68) est enrichie d'un indice supplémentaire :

$$\forall\left(t,t'\right)\in\{1,\cdots,T\}^2,\,\forall i\in\{1,\cdots,N\},$$
$$\alpha_{t,t'}^{H_i}(q)=\begin{cases}\mathbb{P}\left(O_t,\ldots,O_{t'},L_t,\ldots,L_{t'},q_{t'}=q\mid L_i\right)\;\text{ si }t\leqslant t'\\ 0\text{ sinon}\end{cases} \tag{4.74}$$

Le calcul de (4.74) s'effectue grâce à un algorithme forward (E.2.4). Il est possible d'exprimer la suite (4.72) à partir des suites (4.69) et (4.74) comme suit :

$$\forall t\in\{1,\cdots,T\},\,\forall i\in\{1,\cdots,N\},\,\forall q\in H_i,\;\;\alpha_t^{H_i}(q)=\sum_{k=1}^{t-1}\gamma_k^{H_{i-1}}\,\alpha_{k+1,t}^{H_i}(q) \tag{4.75}$$

Le calcul de la suite (4.73) nécessite l'introduction de nouvelles suites, tout d'abord :

$$\forall\left(t,t'\right)\in\{1,\cdots,T\}^2,\,\forall i\in\{1,\cdots,N\},$$
$$\beta_{t,t'}^{H_i}(q)=\begin{cases}\mathbb{P}\left(O_t,\ldots,O_{t'},L_t,\ldots,L_{t'},\mid q_{t-1}=q,\,L_i\right)\;\text{ si }t\leqslant t'\\ 0\text{ sinon}\end{cases} \tag{4.76}$$

Et :

$$\begin{aligned}\forall i\in\{1,\cdots,N\},\quad\delta_{T+1}^{H_i}&=1\\ \forall i\in\{1,\cdots,N\},\,\forall t\in\{1,\cdots,T\},\quad\delta_t^{H_i}&=\sum_{k=t+1}^{T+1}\delta_k^{H_{i+1}}\,\alpha_{t,k}^{H_i}\end{aligned} \tag{4.77}$$

Par conséquent :

$$\forall t \in \{1, \cdots, T\}, \forall i \in \{1, \cdots, N\}, \forall q \in H_i, \quad \beta_t^{H_i}(q) = \sum_{k=t+1}^{T} \delta_k^{H_{i+1}} \alpha_{t,k}^{H_i}(q) \tag{4.78}$$

En tenant compte qu'un mot peut contenir plusieurs fois la même lettre et donc qu'un coefficient peut faire partie de plusieurs modèles H_i, il est possible, armé de toutes ces suites, d'adapter les formules établies dans la table E.2 (voir table 4.17).

$$\overline{\pi_{q,d}^{H_l}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \gamma_t^{k,H_{i-1}} \pi_{q,d}^{H_i} \delta_d(L_t)\, b_q^{k,H_i}\left(O_t^k\right) \beta_t^{k,H_i}(q)}{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \gamma_t^{k,H_{i-1}} \delta_t^{k,H_i}} \tag{4.79}$$

$$\overline{a_{q,q',d}^{H_l}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, a_{q,q',d}^{H_l} \delta_d\left(L_t^k\right) b_{q'}^{k,H_i}\left(O_{t+1}^k\right) \beta_{t+1}^{k,H_i}(q')}{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, \beta_t^{k,H_i}(q)} \tag{4.80}$$

$$\overline{\theta_q^{H_l}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, \theta_q^{H_l} \delta_t^{k,H_i}}{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, \beta_t^{k,H_i}(q)} \tag{4.81}$$

émissions discrètes (voir table E.2)

$$\overline{b_q^{H_l}(o)} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \mathbb{1}_{\{O_t=o\}}\, \alpha_t^{k,H_i}(q)\, \beta_t^{k,H_i}(q)}{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, \beta_t^{k,H_i}(q)} \tag{4.82}$$

émissions continues (voir table E.3)

$$\overline{c_{q,c}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \frac{\alpha_t^{k,H_l}(q)\, c_{q,c}\, \beta_t^{k,H_l}(q)\, \mathbb{P}\left(O_t^k \mid c\right)}{\sum_{d=1}^{N} \mathbb{P}\left(O_t^k \mid d\right) c_{q,d}}}{\sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbb{1}_{\{H_l=H_i^k\}} \sum_{t=1}^{T_k} \alpha_t^{k,H_i}(q)\, \beta_t^{k,H_i}(q)} \tag{4.83}$$

Tab. 4.17: Formules de réestimation de Baum-Welch pour des modèles de lettres, les suites utilisées sont celles définies en (4.69), (4.74), (4.77), (4.76). Elles sont valables pour un modèle de lettre noté H_l. L'écriture $\mathbb{1}_{\{H_l=H_i\}}$ est la fonction qui retourne 1 lorsque le modèle H_i est associé à la lettre l. La notation $s(q)$ désigne les états qui suivent l'état q, le coefficient $a_{q,q'}^{H_i}$ désigne la probabilité de transition de l'état q à l'état q' du modèle H_i où $q' \in s(q)$, $b_{q'}^{k,H_i}(O_{t+1})$ est la probabilité d'émission de l'observation O_{t+1}^k par l'état q' du modèle H_i.

La base d'apprentissage est une liste des séquences d'observations $\left(O^1,\dots,O^K\right)$, $\left(L^1,\dots,L^K\right)$ où chaque séquence est égale à $O^k = \left(O^k_1,\dots,O^k_{T_k}\right)$ et $L^k = \left(L^k_1,\dots,L^k_{T_k}\right)$, ces séquences d'observations O^k et de liaisons L_k correspondent au mot $m_k = \left(l^k_1,\dots,l^k_{N_k}\right)$ dont le modèle associé est le modèle $M_k = \left(H^k_1,\dots,H^k_{N_k}\right)$. La réestimation des paramètres des modèles de lettre utilise la définition des suites (4.69), (4.74), (4.77), (4.76) surmontées d'un indice supplémtenaire (k) correspondant aux séquences d'observations O^k de liaisons L^k à partir desquelles elles ont été calculées.

En pratique, les dénominateurs des formules de la table 4.17 ne sont pas calculés sauf pour vérifier que le calcul des numérateurs est correct. Il est préférable de renormaliser en utilisant les contraintes sur les coefficients. Dans le cas contraire, à cause de la précision de calcul des ordinateurs et le grand nombre d'opérations, les contraintes sont de moins en moins vérifiées lorsque le nombre d'itérations de l'algortihme EM s'accroît.

L'apprentissage des réseaux de neurones associés aux observations et aux liaisons est identique à la méthode décrite au paragraphe 4.5.3 à la différence que la réestimation des sorties désirées pour ces réseaux de neurones fait intervenir non plus un modèle de lettre mais un modèle de mot.

4.6.4 Comparaison HMM - IOHMM

L'apport des modèles IOHMM par rapport aux modèles de Markov cachés a été mesuré sur une base de 2000 images de mots anglais couramment employés [29]. Ces 2000 mots ont été choisis aléatoirement pour un dictionnaire de 500 mots. 1000 servent à l'estimation tandis que les 1000 autres forment la base de test. Un premier modèle de reconnaissance est construit sans tenir compte des liaisons et est estimé sur cette base. Ce modèle est constitué d'un jeu de vingt-six modèles de lettres dont le nombre d'états est défini selon la règle décrite au paragraphe 4.5.4. Le réseau de neurones classant les graphèmes contient vingt-six classes de sortie et dix neurones sur la couche cachée. Les caractéristiques utilisées sont $Mat(5,5)$.

L'expérience continue en reprenant ce même modèle, chaque transition est ensuite dupliquée à l'identique autant de fois qu'il y a de classes de liaisons. Un réseau de neurones classant les liaisons est ajouté, il possède quatre sorties et quatre neurones sur la couche cachée. Le système est réappris jusqu'à convergence. Les résultats sont donnés dans la table 4.18.

base	HMM	IOHMM
apprentissage	43,4 %	52,4 %
test	42,6 %	48,2 %

Tab. 4.18: Comparaison des modèles HMM (Hidden Markov Model) et IOHMM (Input Output Hidden Markov Model). Le modèle IOHMM est construit à partir du modèle HMM en ajoutant les connexions relatives aux liaisons. Les chiffres obtenus correspondent au taux de reconnaissance. Le faible écart entre les bases d'apprentissage et de test suggère qu'il n'y a pas eu de surapprentissage. L'écart est néanmoins plus grand dans le cas du modèle IOHMM qui contient plus de coefficients.

L'apport des liaisons est plus important pour les modèles de Markov cachés que pour la reconnaissance effectuée à partir de plus proches voisins au paragraphe 4.4. En effet, lors de la recherche des plus proches voisins, la distance entre deux images est la somme de deux distances entre séquences de liaisons et entre séquences d'observations. Dans ce cas, les liaisons sont inséparables des graphèmes. En revanche, lors de l'apprentissage des IOHMM, les coefficients relatifs aux liaisons accumulent les informations indépendamment des observations.

29. Cette base est extraite de celle donnée à l'adresse *http ://www.cs.nott.ac.uk/∼dge/words.tar.gz* sur le site ICDAR 2003 (voir [ICDAR2003]).

Remarque 4.6.6: bases difficiles
Les systèmes de reconnaissance incluent un grand nombre de paramètres si bien que la vraisemblance définit un grand nombre de minima locaux, d'autant plus grand que les bases d'apprentissage sont difficiles - un grand nombre d'images dont l'écriture est peu lisible -. Pour contourner ce problème, il est préférable d'isoler une partie réduite de la base d'apprentissage pour laquelle les images sont de bonnes qualité dont la reconnaissance ne pose pas de problèmes majeurs. Après la convergence des modèles de reconnaissance effectuée à partir de cette base réduite, des exemples plus difficiles sont régulièrement ajoutés à la base d'apprentissage lors de la réestimation des coefficients jusqu'à inclure l'ensemble de la base initiale. Dans certains cas, ce procédé a permis d'obtenir des performances acceptables que ne pouvait atteindre un apprentissage utilisant constamment l'ensemble de la base et qui n'a pas permis d'éviter le piège des nombreux minima locaux introduit par le grand nombre d'images bruitées.

4.7 Modélisation de groupes de lettres

4.7.1 Présentation

Les modèles de mot présentés dans le paragraphe 4.6.2 supposent que la segmentation en graphèmes est correcte, qu'aucun graphème ne regroupe ensemble les morceaux de plus d'une lettre. Cette hypothèse est loin d'être toujours vérifiée. Les travaux présentés dans l'article [Chen1994] sont fondés sur un prétraitement qui ne peut segmenter le couple "TT" dans 75% des cas. Il apparaît donc que certains couples de lettres voire certains groupes de lettres sont couramment mal segmentés (figure 3.2, page 41). Certains types d'erreurs sont récurrents :

1. les accents et les poids associés à une lettre voisine de la bonne lettre
2. les liaisons hautes introduites par les lettres b,o,v,w qui impliquent une déformation de la lettre suivante
3. les lettres hautes qui altèrent localement la segmentation (barre de t, double l, ...)

La proportion d'erreurs est difficile à évaluer autrement que manuellement. Toutefois, la méthode proposée dans ce paragraphe permet de tenir compte des erreurs les plus fréquentes et d'en donner une estimation. Ce paragraphe concerne la poursuite des travaux décrits dans [Dupré2002] et plus récemment [Dupré2004].

Prenons l'exemple du mot "attention", les groupes de lettres "tt", "ti", "on", "tion" sont parfois mal segmentés. Même si le groupe "tion" contient plus de deux lettres, il peut parfois apparaître au travers d'un seul graphème. Ce groupe a été ajouté de manière à montrer que cette méthode ne se limite pas seulement aux couples de lettres mal segmentés. La liste des modèles de lettres disponibles est formée des vingt-six lettres usuelles et des quatre groupes de lettres précédemment cités. L'extension de l'alphabet de modèles aboutit à dix manières différentes d'écrire le mot "attention" (table 4.19). Ces dix manières peuvent être résumées en un seul graphe (figure 4.32) qui décrit le squelette sur lequel le modèle du mot "attention" s'appuie. Ce modèle sera noté $G(m)$ (ou modèle graphe).

a,t,t,e,n,t,i,o,n	a,tt,e,n,t,i,o,n
a,t,t,e,n,t,i,on	a,tt,e,n,t,i,on
a,t,t,e,n,ti,o,n	a,tt,e,n,ti,o,n
a,t,t,e,n,ti,on	a,tt,e,n,ti,on
a,t,t,e,n,tion	a,tt,e,n,tion

Tab. 4.19: Dix manières différentes d'écrire le mot "attention" en utilisant les vingt-six modèles de l'alphabet et les quatre groupes de lettres "tt", "ti", "on", "tion".

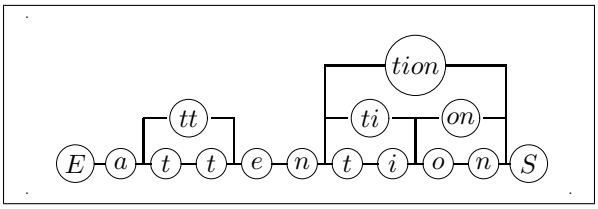


Fig. 4.32: Modèle pour le mot "attention", ce graphe résume les dix manières différentes d'écrire ce mot d'après la table 4.19. Il n'existe pas de graphe plus concis que celui noté $G(m)$ et représenté par ce schéma.

Cette modélisation rejoint celle décrite dans l'article [Wang2000]. Ce dernier s'intéresse à la modélisation de toutes les paires de chiffres qui sont presque toutes équiprobables - la paire 00 est généralement plus probable que les autres quand il s'agit de montant -. En ce qui concerne les caractères, les lettres ne sont déjà pas équiprobables, les couples de lettres encore moins, il est donc impossible de modéliser tous les couples sachant que certains ne peuvent de toutes façons pas apparaître (le couple "ZZ").

4.7.2 Probabilité

Il reste maintenant à calculer la probabilité $\mathbb{P}(O, L \mid G(m))$ qu'un tel modèle émette une séquence d'observations. Soit m un mot et $\{l_1, \cdots, l_N\}$ le plus grand ensemble de lettres permettant d'écrire le mot m de toutes les manières possibles. L'ensemble $\{H_1, \cdots, H_N\}$ correspond aux modèles de Markov cachés : la lettre l_i est modélisée par le modèle H_i.

Le calcul de $\mathbb{P}(O, L \mid G(m))$ nécessite l'introduction de la matrice A et des vecteurs Π et Θ :

- Le vecteur $\Pi = (\pi_i)_{1 \leqslant i \leqslant N}$, π_i étant la probabilité de commencer par la lettre i.
- La matrice $A = (a_{ij})_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant N}}$, a_{ij} étant la probabilité d'atteindre la lettre j en partant de la lettre i.
- Le vecteur $\Theta = (\theta_i)_{1 \leqslant i \leqslant N}$, θ_i étant la probabilité de terminer le mot m par la lettre i.

La table 4.20 donne ces trois éléments pour le mot "attention". Ces coefficients sont estimés de manière similaire à ceux d'une chaîne de Markov dont les états sont les lettres. Par exemple :

$$a_{ij} = \frac{\text{nombre de chemins contenant les lettres } l_i \text{ et } l_j}{\text{nombre de chemins contenant la lettre } l_i}$$

Ces coefficients vérifient les contraintes suivantes :

$$\sum_{i=1}^{N} \pi_i = 1 \text{ et } \forall i, \sum_{j=1}^{N} a_{ij} + \theta_i = 1$$

Les suites $\left(\alpha_{t,t'}^{H_i}\right)_{i,t,t'}$ et $\gamma_t(H_i)$ sont presque identiques à celles définies en (4.68) et (4.69).

$$\begin{aligned} \forall i \ \gamma_1(L_i) &= \pi_{H_i} \alpha_{1,1}^{H_i} \\ \forall i \ \forall t \geqslant 2, \ \gamma_t(H_i) &= \pi_{H_i} \alpha_{1,t}^{H_i} + \sum_{j=1}^{N} \sum_{k=1}^{t-1} \gamma_{t-k}(H_j) \, a_{l_j, l_i} \, \alpha_{t-k+1,t}^{H_j} \end{aligned} \tag{4.84}$$

L'expression de $\mathbb{P}(O \mid G(m))$ devient :

i	1	2	3	4	5	6	7	8	9	1.	11	12	13
l_i et L_i	a	t	t	e	n	t	i	o	n	tt	ti	on	tion
π_i	1	.	.	.	.	.	.	.	.	.	.	.	.
θ_i	.	.	.	.	.	.	.	.	1	.	.	1	1
a_{1i} (a →)	.	0,5	.	.	.	.	.	.	.	0,5	.	.	.
a_{2i} (t →)	.	.	1	.	.	.	.	.	.	.	.	.	.
a_{3i} (t →)	.	.	.	1	.	.	.	.	.	.	.	.	.
a_{4i} (e →)	.	.	.	.	1	.	.	.	.	.	.	.	.
a_{5i} (n →)	.	.	.	.	.	0,4	.	.	.	.	0,4	.	0,2
a_{6i} (t →)	.	.	.	.	.	.	1	.	.	.	.	.	.
a_{7i} (i →)	.	.	.	.	.	.	.	0,5	.	.	.	0,5	.
a_{8i} (o →)	.	.	.	.	.	.	.	.	1	.	.	.	.
a_{9i} (n →)	.	.	.	.	.	.	.	.	.	.	.	.	.
a_{10i} (tt →)	.	.	.	1	.	.	.	.	.	.	.	.	.
a_{11i} (ti →)	.	.	.	.	.	.	.	0,5	.	.	.	0,5	.
a_{12i} (on →)	.	.	.	.	.	.	.	.	.	.	.	.	.
a_{13i} (tion →)	.	.	.	.	.	.	.	.	.	.	.	.	.

Tab. 4.20: Matrice A et vecteurs Π, Θ pour le mot "attention".

$$\mathbb{P}\left(O, L \mid G\left(m\right)\right) = \sum_{i=1}^{N} \theta_{H_i} \, \gamma_T\left(H_i\right) \tag{4.85}$$

Le calcul de (4.85) est plus coûteux que (4.70). Cependant, il est possible de tenir compte des nombreux coefficients nuls des matrices A, Π, Θ afin de réduire le coût de l'algorithme suivant 4.7.1.

Algorithme 4.7.1 : probabilité d'une séquence avec un modèle de mot
Etape A : initialisation 1
Calcul de la suite $\left(\alpha_{t,t'}^{H_i}\right)_{t,t',i}$ grâce à l'algorithme E.2.3.

Etape B : initialisation 2
pour $t = 1$ **à** T **faire**
pour $i = 1$ **à** N **faire**
$\gamma_t^{L_i} \longleftarrow \pi_{H_i} \, \alpha_{1,1}^{H_i}$
fin pour
fin pour

Etape C : itérations
pour $t = 1$ **à** T **faire**
pour $i = 1$ **à** N **faire**
pour $k = 1$ **à** $t - 1$ **faire**
pour $j = 1$ **à** N **faire**
$\gamma_t^{H_i} \longleftarrow \gamma_t^{H_i} + \gamma_k^{H_j} \, a_{l_j,l_i} \, \alpha_{k+1,t}^{H_i}$
fin pour
fin pour
fin pour
fin pour

Etape D : terminaison
$p \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$p \longleftarrow \theta_{H_i} \, \gamma_T\left(H_i\right)$
fin pour
La probabilité cherchée est p.

Cette modélisation se rapproche d'un modèle de Markov caché dont les états acceptent des émissions

de durées variables. Les probabilités $\alpha_{t,t'}(.)$ peuvent être considérées comme les émissions d'une chaîne de Markov cachée définie par les coefficients A, Π, Θ. Ces modèles sont plus souvent utilisés pour la reconnaissance de la parole ([Mitchell1995]).

4.7.3 Expérimentations

La première expérience est un problème de reconnaissance simplifiée qui s'appuie sur une base d'apprentissage de 1200 mots anglais et une base de test de 800 mots. Ces mots appartiennent tous au dictionnaire composé des mots présentés dans la table 4.21 et ne faisant intervenir que les neuf lettres b, d, e, f, g, i, n, o, p. Huit modèles de reconnaissance, un par lettre, sont donc estimés à partir de ces données et permettent d'obtenir les taux de reconnaissance de la table 4.22.

BE	DEED	FIND	INDEPEND
BEEN	DEEP	FINE	NEED
BEGIN	DEFEND	FINDING	NEEDED
BEING	DEFINE	FOND	NEEDING
BIN	DEPEND	FOOD	NINE
BED	DID	GIG	NO
BEND	DIE	GOOD	ONE
DEED	DOING	I	OPEN
END	DONE	ID	OPENED
DEPEND	EGG	IN	OPENING
BOND	ENDED	INDEED	PEN

Tab. 4.21: Dictionnaire réduit n'utilisant que les neuf lettres b, d, e, f, g, i, n, o, p.

	taux de reconnaissance	taux de reconnaissance du mot BEING
apprentissage	70,8 %	70,0 %
test	62,3 %	57,5 %

Tab. 4.22: Les performances en reconnaissance sont mesurés sur les base d'apprentissage (1200 mots) et de test (800 mots). Elles sont sensiblement moins élevés sur la base de test car étant donné la petite taille des bases, les modèles ont tendance à "sur-apprendre". Ces bases ont été construites de telles sortes que le mot BEING représente un dixième de ces deux bases.

La même expérience est recommencée en incluant cette fois-ci dans l'alphabet le groupe de lettres ING très fréquent en langue anglaise et, pour cette raison, parfois mal écrit. Les résultats apparaissent dans la table 4.23.

	taux de reconnaissance	taux de reconnaissance du mot BEING
apprentissage	72,8 %	86,6 %
test	64,9 %	71,2 %

Tab. 4.23: Même taux de reconnaissance que celui de la table 4.22 mesuré cette fois avec un alphabet incluant le groupe de lettres supplémentaire ING.

Il apparaît dans la première expérience que le taux de reconnaissance du mot BEING est équivalent au taux de reconnaissance sur l'ensemble de la base. En revanche, lors de la seconde expérience, un modèle à associé au groupe ING. Le taux de reconnaissance du mot BEING est alors nettement plus important

que dans la première expérience et explique à lui seul l'accroissement du taux de reconnaissance global. Ces deux expériences montrent l'intérêt de la modélisation des groupes de lettres souvent mal segmentés.

L'apport de ces modèles a ensuite été testé sur une base réelle de 40000 prénoms français, 30000 ont servi à l'estimation des modèles, 10000 aux tests. La table 4.24 compare les performances obtenus en utilisant ou non les modèles de bi-lettres. Elle montre l'accroissement des performances pour un alphabet étendu. Le taux de reconnaissance [30] ne croît pas de manière significative mais le taux de lecture pour 1% de confusion [31] a augmenté de manière sensible. La liste des modèles de lettres et de bi-lettres utilisés pour cette expérience est recensée par la table 4.25.

	base	taux de reconnaissance	taux de lecture pour 1% de confusion	seuil
sans bi-lettres	app	88,8 %	60,0 %	0,976
	test	88,7 %	61,9 %	0,980
avec bi-lettres	app	90,0 %	65,3 %	0,972
	test	89,0 %	65,7 %	0,968

Tab. 4.24: Evaluation des performances des modèles sans bi-lettres et avec bi-lettre.

4.7.4 Segmentation la plus probable

L'alignement Viterbi [32] ([Levinson1983], [Rabiner1986]) est un algorithme permettant d'obtenir la séquence d'états la plus probable connaissant la séquence d'observations. Il peut être adapté à la recherche de la séquence de modèles de lettre la plus probable. Appliqué au mot "attention", l'objectif est de déterminer quelle écriture est la plus vraisemblable parmi les dix possibles (table 4.19). Cette meilleure séquence, si elle inclut des modèles de groupes de lettres, pourrait permettre de détecter ces erreurs de segmentation. Elle ne peut en revanche pas détecter des erreurs de segmentation concernant un couple de lettres non modélisé.

Par conséquent, pour des séquences d'observations et de liaisons données $O = (O_1, \ldots, O_T)$ et $L = (L_1, \ldots, L_T)$, on cherche une séquence de lettres (et groupes de lettres) notée $(H_1^*, \ldots, H_U^*)$ avec $U \leqslant N$. Pour chaque modèle, H_i^*, $\delta_{H_i}^*$ désigne le nombre d'observations qui lui sont associées.

$$\forall i \in \{1, \cdots, U\},\ d_{H_i^*}^* = \begin{cases} 0 \text{ si } i = 1 \\ 1 + \sum_{k=1}^{i-1} \delta_{H_i^*}^* \text{ sinon} \end{cases}$$

D'où :

$$\mathbb{P}\left(O_1, \ldots, O_T, L_1, \ldots, L_T, H_1^*, \ldots, H_U^*, \delta_{H_1^*}^*, \ldots, \delta_{H_U^*}^* \mid G(m)\right) = \pi_{H_1^*} \alpha_{1,\delta_{H_1^*}^*}^{H_1^*} \left[\prod_{i=2}^{U} a_{H_{i-1}^*, H_i^*} \alpha_{d_{H_i^*}^*, d_{H_i^*}^* + \delta_{H_i^*}^* - 1}^{H_i^*}\right] \theta_{H_U^*} \quad (4.86)$$

La probabilité de toute autre séquence de modèles sera inférieure à (4.86). Il reste à trouver cette meilleure séquence $(H_1^*, \ldots, H_U^*)$. On utilise pour cela les suites $(p_{t,H_i})_{t,i}$, $(m_{t,H_i})_{t,i}$, $(s_{t,H_i})_{t,i}$:

30. Le taux de reconnaissance est la proportion de documents bien reconnus.

31. Le taux de lecture pour 1% de confusion correspond à la proportion de documents qui peuvent être traités avec un taux d'erreur inférieur à 1%. Ces taux sont détaillés au parapgraphe 5.1.

32. Annexes : voir paragraphe E.3, page 260

– p_{t,H_i} mémorise la probabilité de la meilleure séquence de lettres se terminant par la lettre H_i à l'instant t.
– s_{t,H_i} mémorise le nombre d'observations associées au modèle H_i qui a permis d'obtenir la meilleure probabilité p_{t,H_i}
– m_{t,H_i} mémorise le modèle précédent L_i à l'instant $t - s_{t,H_i}$ qui a permis d'obtenir la meilleure probabilité p_{t,H_i}

L'initialisation de ces suites est donnée par les formules qui suivent :

$$\begin{array}{rcl} \forall i \in \{1,\cdots,N\}, \forall t \in \{1,\cdots,T\}, \quad p_{t,H_i} & = & \pi_{H_i}\, \alpha_{1,t}^{H_i} \\ \forall i \in \{1,\cdots,N\}, \forall t \in \{1,\cdots,T\}, \quad s_{t,H_i} & = & t \\ \forall i \in \{1,\cdots,N\}, \forall t \in \{1,\cdots,T\}, \quad m_{l,H_i} & = & -1 \end{array} \tag{4.87}$$

Le calcul se poursuit en faisant croître t :

$$\begin{array}{rcl} \forall i \in \{1,\cdots,N\}, \forall t \in \{2,\cdots,T\}, \quad (s_{t,H_i}, m_{t,H_i}) & = & (j^*,t^*) \in \underset{\substack{1\leqslant j\leqslant N \\ 1\leqslant k\leqslant t-1}}{\arg\max}\, p_{t-k,H_j}\, a_{H_j,H_i}\, \alpha_{t-k+1,t}^{H_i} \\ \forall i \in \{1,\cdots,N\}, \forall t \in \{2,\cdots,T\}, \quad p_{t,H_i} & = & p_{t-m_{t,H_i},L_{s_{t,H_i}}}\, a_{H_{s_{t,H_i}},H_i}\, \alpha_{t-m_{t,H_i}+1,t}^{H_i} \end{array} \tag{4.88}$$

Et finalement :

$$\begin{array}{rcl} (m_{T+1}) & = & j^* \in \underset{1\leqslant j\leqslant N}{\arg\max}\, p_{T,H_j}\, \theta_{H_j} \\ p_{T+1} & = & p_{T,H_{s_{T+1}}}\, \theta_{H_{s_{T+1}}} \end{array} \tag{4.89}$$

La probabilité de la meilleure séquence de lettres est donnée par p_{T+1}, les suites $(s_{t,H_i})_{t,i}$ et $(m_{t,H_i})_{t,i}$ permettent de retrouver cette séquence puisqu'elles conservent pour chaque maximum local les élements

qui ont permis de l'obtenir. Les formules (4.87), (4.88), (4.89) aboutissent à l'algorithme suivant :

Algorithme 4.7.2 : meilleure séquence de lettres (1)
Cet algorithme permet de calculer les suites $(p_{t,H_i})_{t,i}$, $(m_{t,H_i})_{t,i}$, $(s_{t,H_i})_{t,i}$.
Etape A : initialisation 1
Calcul de la suite $\left(\alpha_{t,t'}^{H_i}\right)_{t,t',i}$ grâce à l'algorithme E.2.3.

Etape B : initialisation 2
pour $i = 1$ **à** N **faire**
pour $t = 1$ **à** T **faire**
$p_{t,H_i} \longleftarrow \pi_{H_i}\, \alpha_{1,t}^{H_i}$
$m_{t,H_i} \longleftarrow i$
$s_{t,H_i} \longleftarrow -1$
fin pour
fin pour

Etape C : récurrence
pour $i = 1$ **à** N **faire**
pour $t = 2$ **à** T **faire**
$p_{t,H_i} \longleftarrow 0$
$m_{t,H_i} \longleftarrow -1$
$s_{t,H_i} \longleftarrow -1$
pour $j = 1$ **à** N **faire**
pour $k = 1$ **à** $t-1$ **faire**
$x \longleftarrow p_{t-k,H_j}\, a_{H_j,H_i}\, \alpha_{t-k+1,t}^{H_i}$
si $x > p_{t,H_i}$ **alors**
$p_{t,H_i} \longleftarrow x$
$m_{t,H_i} \longleftarrow j$
$s_{t,H_i} \longleftarrow k$
fin si
fin pour
fin pour
fin pour
fin pour

Etape D : terminaison
$p_{T+1} \longleftarrow 0$
$m_{T+1} \longleftarrow -1$
pour $j = 1$ **à** N **faire**
pour $k = 1$ **à** $t-1$ **faire**
$x \longleftarrow p_{T+1,H_j}\, \theta_{H_j}$
si $x > p_{t,H_i}$ **alors**
$p_{t,H_i} \longleftarrow x$
$m_{t,H_i} \longleftarrow j$
fin si
fin pour
fin pour

La meilleure séquence est finalement obtenue par l'algorithme qui suit.

Algorithme 4.7.3 : meilleure séquence de lettres (2)
A partir des suites $(p_{t,H_i})_{t,i}$, $(m_{t,H_i})_{t,i}$, $(s_{t,H_i})_{t,i}$ calculées à partir de l'algortihme 4.7.2, cet algorithme permet d'obtenir la meilleure séquence de lettres ainsi que les observations parmi la séquence $(O_1, \ldots, O_T)$ qui leur sont associées.
La séquence de modèles est notée $(H_1^*, \ldots, H_U^*)$ et la séquence des nombres d'observations associéss à chaque modèle est notée $(\delta_1^*, \ldots, \delta_U^*)$

Etape A : initialisation

$U \longleftarrow 1$
$H_U^* \longleftarrow m_{T+1}$
$\delta_U^* \longleftarrow s_{T,H_U^*}$
$t \longleftarrow T$
$U \longleftarrow U+1$

Etape B : récurrence

tant que $(H_{U-1}^* \neq -1)$ **faire**
$H_U^* \longleftarrow m_{t,H_{U-1}^*}$
$\delta_U^* \longleftarrow s_{t,H_{U-1}^*}$
$t \longleftarrow t - \delta_{U-1}^*$
$U \longleftarrow U+1$
fin tant que
$U \longleftarrow U-2$

La séquence obtenue est retournée.

Etape C : terminaison

$i \longleftarrow 1$ et $j \longleftarrow U$
tant que $(i < j)$ **faire**
$H_i^* \longleftrightarrow H_j^*$
$\delta_i^* \longleftrightarrow \delta_j^*$
$i \longleftrightarrow i+1$
$j \longleftrightarrow j-1$
fin tant que

Théorème 4.7.4 : meilleure séquence de lettres
Pour deux séquences d'observations et de liaisons $O = (O_1, \ldots, O_T)$ et $L = (L_1, \ldots, L_T)$, les séquences de modèles et durées $(H_1^*, \ldots, H_U^*)$ et $(\delta_1^*, \ldots, \delta_U^*)$ obtenues par les algorithmes 4.7.2 et 4.7.3 sont celles qui maximisent (4.86) :

$$\mathbb{P}\left(O_1, \ldots, O_T, L_1, \ldots, L_T H_1^*, \ldots, H_U^*, \delta_{H_1^*}^*, \ldots, \delta_{H_U^*}^* \mid G(m)\right)$$

Démonstration (théorème 4.7.4) :

La démonstration est analogue à celle de l'algorithme E.3.1. Pour résumer, celle-ci s'effectue par récurrence sur t, afin de montrer que pour tout (t, i), p_{t,H_i} est la probabilité correspondant à la séquence de modèles la plus probable parmi toutes celles se terminant à l'instant t par le modèle H_i. C'est de manière évidente vrai pour $t = 1$ et ce quel que soit i, il suffit de le montrer pour $t+1$.

(4.7.4) □

La table 4.25 illustre les résultats sur l'utilisation des modèles de bi-lettres. Toutefois, une estimation précise des erreurs de segmentation avec les modèles de bi-lettres est difficilement déductible des résultats présentés dans ce tableau. Les modèles de bi-lettres apprennent non seulement les cas mal segmentés mais aussi les cas bien segmentés. Pour contrecarrer ce penchant, il faudrait peut-être empêcher les modèles de bi-lettres de pouvoir émettre les mêmes classes d'observations que celles qu'émettent les modèles associées aux lettres qui le composent. Par exemple, soit i un état quelconque du modèle "LL", le coefficient $c_{i,c}^{LL}$ associé à l'émission d'une observations de la classe c pourrait être diminué lors de sa mise à jour si $\max_j c_{j,c}^{L}$ est non négligeable. Cette pénalisation n'est néanmoins pas aussi simple à mettre en œuvre puisque l'estimation des coefficients des modèles IOHMM est une optimisation sous contraintes.

L'algorithme permettant de trouver la meilleure séquence de modèles de lettres ne donne pas de résultats réellement satisfaisants lorsqu'elle est appliquée à la détection des erreurs de segmentation. Une meilleure détection nécessiterait de plus amples développements. Toutefois, le paragraphe 4.8 montrera que l'algorithme de Viterbi permet d'obtenir des résultats intéressants.

Remarque 4.7.5: segmentations les plus probables
Les algorithmes 4.7.2 et 4.7.3 permettent d'obtenir la meilleure séquence de modèles mais il possible d'obtenir la liste des séquences les plus probables en conservant à chaque itération t et pour chaque modèle i des informations non plus sur le meilleur chemin mais sur les n meilleurs chemins. L'article [Chen1994] présente une version optimisée permettant d'obtenir les n meilleures séquences.

4.7.5 Pour aller plus loin, génération de caractères manuscrits

L'idée s'inspire de l'article [ChoiH2003] qui propose la génération aléatoire de caractères manuscrits à partir de modèles écrits. Les modèles de Markov cachés permettent de générer aléatoirement des séquences probables d'états puis des séquences de vecteurs de probabilité de modèles et enfin des séquence de caractéristiques.

Pour peu que les séquences de caractéristiques permettent de fabriquer une image représentative du caractère - c'est le cas des caractéristiques décrites aux paragraphes 4.2.3, 4.2.7, 4.2.8, 4.2.9 -, il est possible pour un mot donné de générer quelques écritures les plus probables. Cela permet d'obtenir une représentation visuelle de ce que les modèles de reconnaissance ont appris. Cette méthode pourrait être utilisée pour mieux visualiser les différentes formes de lettres apprises par un modèle de bi-lettres.

4.7.6 Apprentissage

L'apprentissage des modèles de lettres et groupes de lettres utilisent les mêmes formules que celles de la table 4.17 mais avec des suites $\left(\alpha_t^{k,H_i}(q)\right)$, $\left(\beta_t^{k,H_i}(q)\right)$, $\left(\gamma_t^{k,H_i}\right)$, $\left(\delta_t^{k,H_i}\right)$ différentes mais facilement déductibles de celles définies par les équations (4.72) à (4.78). L'annexe G propose un formalisme fondé sur les graphes permettant d'exprimer avec plus de clarté aussi bien le calcul des probabilités que la mise à jour des paramètres des modèles de Markov caché.

4.8 Reconnaissance sans dictionnaire

La reconnaissance sans dictionnaire utilise le même formalisme que celui développé au paragraphe 4.7. La reconnaissance sans dictionnaire de deux séquences d'observations et de liaisons $O = (O_1, \ldots, O_T)$ et $L = (L_1, \ldots, L_T)$ est la recherche de la meilleure séquence de lettres dans un modèle particulier, soit l'application des algorithmes 4.7.2 et 4.7.3 au modèle $G(m)$ suivant :

A	24328	83%	M	9630	98%
AI	872	44%	N	20077	70%
AN	7194	47%	O	8490	58%
B	2203	62%	OI	888	44%
BE	1556	44%	OM	252	64%
BI	51	61%	ON	1862	46%
BR	304	39%	OR	804	42%
BU	1	100%	OS	1157	56%
C	8355	67%	OT	65	72%
CE	1837	86%	OU	1487	72%
CH	1888	60%	P	3740	100%
D	5913	74%	Q	764	100%
DE	1975	78%	R	19782	50%
E	46530	59%	RE	3622	61%
ER	4545	64%	RI	6908	62%
ES	2322	58%	S	9555	42%
F	1928	100%	SA	391	68%
G	4384	100%	SE	3016	35%
H	6204	82%	SO	163	42%
I	22890	42%	T	8226	82%
IE	7603	34%	TE	2036	37%
IN	3708	49%	TT	1224	29%
IS	3656	58%	U	8812	82%
J	5083	60%	V	1775	63%
JE	2612	60%	VE	679	47%
JU	952	51%	VI	736	47%
K	247	100%	VU	1	100%
L	15702	69%	W	99	100%
LE	4484	52%	X	235	100%
LI	2633	53%	Y	1689	100%
LL	821	70%	Z	356	100%

Tab. 4.25: Occurrences des lettres et couples de lettres de la base d'apprentissage, la troisième colonne est obtenue grâce à un alignement Viterbi qui détermine quelle écriture est la plus probable. Par exemple lorsque la séquence AI intervient dans un mot, dans 44 % des cas, le modèle associé au couple AI est plus probable que le couple de modèles associés aux lettres A et I. En revanche, dans le cas de la lettre A, dans 83 % des cas, elle est mieux modélisée par son modèle associé plutôt qu'incluse dans un modèle des couples AI, AN, SA.

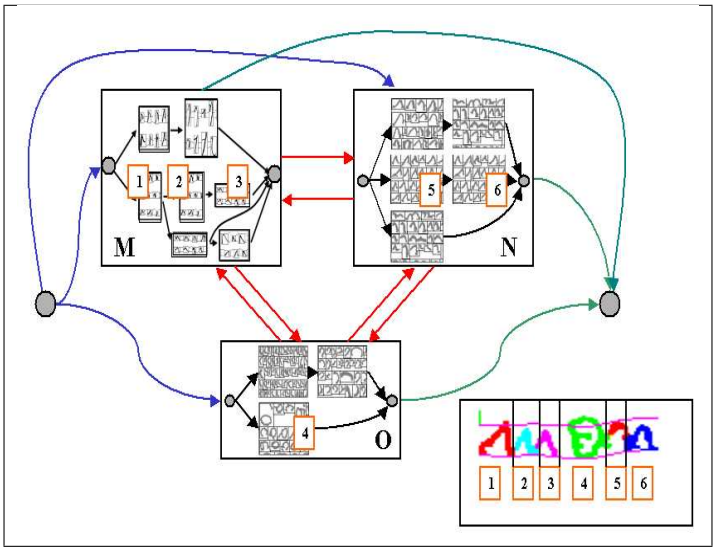


Fig. 4.33: Recherche de la séquence de lettres la plus probable : chaque observation est associée à un état d'un modèle de lettre. La séquence d'états obtenue mène à une séquence de modèles de lettres formant un mot très probable.

– La liste $(H_1, \ldots, H_N)$ des modèles de lettres et groupes de lettres utilisés est l'alphabet entier (étendu ou non).
– Les vecteurs Π, Θ et la matrice A sont uniformes et tous leurs coefficients sont égaux à 1.

Ce principe est illustré pour un alphabet de trois lettres par la figure 4.33. Les performances d'un tel système peuvent être améliorées en estimant les coefficients Π, Θ, A à l'aide de bi-grammes [33]. Inclure des n-grammes avec $n > 2$ est plus délicat et nécessite une version différente des algorithmes 4.7.2 et 4.7.3 et plus coûteuse aussi. Par exemple, toujours sur la même base de 40000 prénoms, cet algorithme permet de trouver le bon prénom dans seulement 5 % des cas. L'utilisation de bi-grammes permet d'augmenter ce pourcentage à 33 %. Cette même expérience a montré que lorsque les reconnaissances avec dictionnaire et sans dictionnaire concordent, le taux d'erreur est nul. Cette remarque sera utilisée au paragraphe 5.3 (page 143).

4.9 Sélection d'architecture

Les algorithmes proposés jusqu'à présent permettent d'estimer les paramètres des modèles de Markov cachés une fois que sa structure a été fixée. Chaque modèle de lettre est initialisé de manière intuitive, le nombre d'états est fonction de la complexité de la forme à représenter et de sa variabilité. Ce problème de sélection de modèles intervient dans de nombreux domaines et s'appuie sur un compromis entre taille de modèle et vraisemblance des observations. Deux critères couramment utilisés sont : *Akaike Information Criterieum* et *Bayesian Information Criterieum* (AIC et BIC) (voir [Akaike1974], [Schwartz1978], [Saporta1990]). Par exemple, le modèle sélectionné AR [34] pour modéliser une série temporelle $(X_1, ..., X_N)$ est celui qui minimise un critère, ici BIC :

33. Annexes : voir paragraphe L, page 415
34. Auto-Regressive

$$BIC\,(d) = \underbrace{\frac{1}{N-d}\sum_{t=d+1}^{N}\left(X_t - \sum_{i=1}^{d}\alpha_i\,X_{t-i}\right)^2}_{\text{erreur moyenne de prédiction}} + \underbrace{d\ln\left(N-d\right)}_{\text{pénalisation}}$$

L'erreur de prédiction décroît uniformément lorsque d augmente mais la faculté de généralisation du modèle diminue : si la taille du modèle est trop importante, son erreur de prédiction est susceptible d'augmenter sur de nouvelles données.

Cette méthode ne peut malencontreusement pas s'adapter telle quelle aux modèles de Markov cachés. Les coefficients ne jouent pas tous le même rôle, certains sont des probabilités d'émissions, d'autres de transitions mais surtout, ces coefficients sont liés entre eux par des contraintes. Si les transitions sont utilisées à chaque séquence d'observations, en revanche, selon cette séquence, les probabilités d'émissions ne sont pas toutes utiles. De plus, un modèle de mot est un assemblage de modèles de lettre, comment déterminer les modèles de lettre qui contiennent trop ou pas assez de coefficients. Il existe également des modèles dont les nombres de coefficients sont différents et qui pourtant sont équivalents en terme de probabilité (voir [Balasubramanian1993], [Kamp1985]). L'article [Ziv1992] propose une méthode permettant de déterminer une chaîne de Markov non cachée optimale mais il faut pour cela estimer les modèles pour toutes les tailles. L'article [Bicego2003] est celui qui s'approche le plus de la tâche à résoudre. Il s'applique sur des modèles de Markov cachés et dans le cadre de la reconnaissance de l'écriture. Il étend la méthode utilisant le critère BIC et enfin, propose un moyen d'optimiser les nombreuses réestimations de coefficients pour des tailles différentes de modèles.

On suppose que la base d'apprentissage est composée des K séquences d'observations $\left(O^1,\ldots,O^K\right)$, $\mathbb{P}\left(O^1,\ldots,O^K\right)$ est donc la vraisemblance des modèles de reconnaissance, N_k est le nombre de paramètres libres pour le modèle d'indice k et n représente la somme des longueurs des séquences d'observations. Le meilleur modèle maximise le critère suivant :

$$BIC\,(k) = \ln\mathbb{P}\left(O^1,\ldots,O^K\right) - \frac{N_k}{2}\ln n \tag{4.90}$$

Les travaux de [Durand2003] montrent que le critère BIC est un des plus pertinents lorsqu'il s'agit de déterminer le nombre d'états de modèles de Markov cachés. Plutôt que d'essayer tous les nombres d'états possibles, l'article [Bicego2003] propose de partir d'un nombre d'états surestimant le nombre d'états réels puis de faire décroître ce nombre d'états en supprimant à chaque étape l'état le moins probable [35]. L'article montre que cette méthode est bien meilleure que de repartir d'une initialisation aléatoire. La direction de recherche choisie consiste à faire évoluer petit à petit l'architecture des modèles de reconnaissance selon les méthodes développées dans l'annexe F. La taille des modèles va osciller en alternant des méthodes de suppression [36] des coefficients et des méthodes de multiplication [37] de ces mêmes coefficients. Cette succession d'étapes permet d'obtenir une série de modèles dont le meilleur est déterminé par le critère BIC (4.90). L'article [Li2004] propose quant à lui un critère basé sur l'entropie des coefficients et fait également décroître le nombre d'états jusqu'à obtenir un nombre optimal. Sa méthode n'est toutefois applicable que dans le cas d'émissions obéissant à une loi gaussienne [38].

En ce qui concerne la reconnaissance de l'écriture manuscrite, la vraisemblance utilisée pour calculer le critère BIC (4.90) utilise autant de modèles de Markov caché qu'il y a de lettres différentes, contrairement à l'article [Bicego2003] ou à la thèse [Durand2003] qui n'utilise qu'un seul modèle. De plus, ces modèles

35. Annexes : voir paragraphe F.1.3, page 297
36. Annexes : voir paragraphe F.2, page 299
37. Annexes : voir paragraphe F.3, page 312
38. Annexes : voir paragraphe E.5.4, page 285

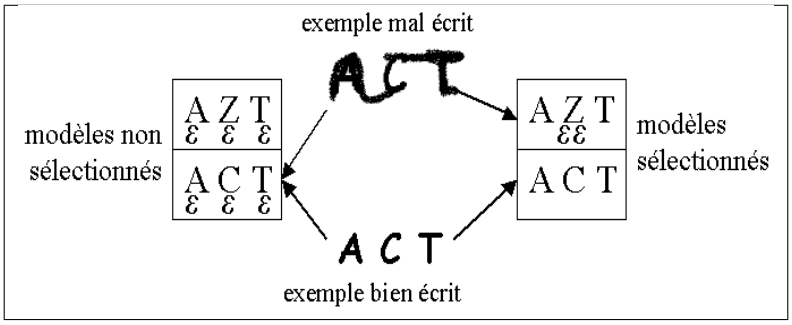


Fig. 4.34: Cette figure illustre de manière schématique les résultats de la reconnaissance pour deux jeux de modèles. Le premier jeu contient des modèles de lettres dont les coefficients n'ont pas été sélectionnés. Chaque modèle de lettre, appris avec beaucoup d'exemples ou non, inclut de nombreuses formes d'écriture symbolisées par le caractère ϵ. Dans ce cas, la reconnaissance, en comparant les modèles "ACT" et "AZT" préfère "ACT" dans les deux cas. Pour la seconde expérience, les lettres "ACT" sont très fréquentes dans la langue française, les modèles ont donc été débarrassés des écritures bruitées alors que le modèle "Z", ayant été estimé avec peu d'exemples, inclut plus d'écritures bruitées. Dans le cas d'une écriture bruitée, la reconnaissance utilisant des modèles sélectionnés choisira le modèle "AZT". Ce schéma est simplifié par rapport à ce qu'on observe dans les expériences mais il donne une bonne intuition des écueils auxquels aboutissent les méthodes de sélection d'architecture.

ne sont pas estimés avec le même nombre d'exemples puisque le modèle de la lettre W française ne peut être estimé qu'avec une centaine de séquences alors que le modèle de la lettre E bénéficie de plusieurs milliers de séquences. Le critère (4.90) intègre donc des coefficients qui ne sont pas tous calculés avec le même nombre d'observations. Les estimations des modèles des lettres moins fréquentes K,Q,W,V,X,Y,Z (voir table 4.25) sont moins fiables et il est difficile de supprimer ou de multiplier un coefficient à choisir dans des modèles de lettres différents à moins de comparer.

Les expériences réalisées dans le cadre de ces travaux ont d'abord montré que les étapes de suppression de coefficients sont moins coûteuses et plus fiables que les étapes de multiplication des coefficients. Il est par conséquent préférable d'estimer un modèle au nombre de coefficients supérieur au nombre optimal. D'autre part, les modèles associés à des lettres fréquentes ont tendance à perdre des coefficients : ils savent idientifier les écritures les plus courantes et rejettent de nombreuses formes aberrantes. Les modèles associés à des lettres moins fréquentes ont tendance à gagner des coefficients : il ne se dégage pas des exemples qu'ils ont à apprendre une écriture courante, en contrepartie, le modèle apprend de nombreuses écritures dont certaines assez bruitées. Lorsque l'écriture devient mal écrite, ces modèles appris avec peu d'exemples et possédant plus de coefficients que les autres deviennent plus probables que ceux appris avec un grand nombre d'exemples (voir figures 4.34 et 4.35).

La recherche d'une méthode de sélection fiable s'articule autour de deux thèmes principaux qui sont une meilleure détermination des coefficients à supprimer et l'amélioration des bases d'apprentissage. Cette seconde option correspond à une préparation des bases d'apprentissage des modèles de façon à gommer les disparités entre lettres comme le suggère le paragraphe 4.3.4. Toutefois, la méthode utilisée pour des caractères doit être adaptée aux mots en tenant compte du fait que dupliquer l'image d'un mot contenant une lettre rare entraîne nécessairement la duplication de lettres fréquentes. La création de bases d'apprentissage plus fiables nécessite a priori de plus amples recherches. La première direction suscite quant à elle toujours l'intérêt des chercheurs comme en témoigne le récent article [Li2004] traitant du problème de la sélection du nombre d'états d'un modèle de Markov caché dont les émissions sont gaussiennes [39]. Comme dans l'article [Bicego2003], le nombre d'états décroît progressivement tout au long de l'apprentissage en choisissant de regrouper ensemble des paires d'états satisfaisant une condition dépendant du nombre de

39. Annexes : voir paragraphe E.5.4, page 285

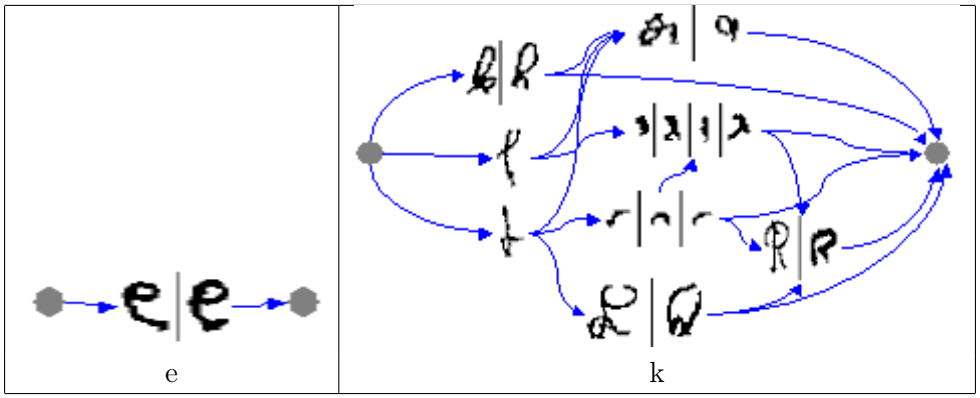


Fig. 4.35: Ces deux modèles de Markov cachés sont une représentation simplifiée de ceux obtenus pour les lettres "E" et "K". Ils illustrent l'idée décrite par la figure 4.34. Chaque classe de graphèmes est représentée par certains de ses éléments les plus probables. Le modèle "e" ne contient plus qu'une seule écriture prépondérante tandis que le modèle "k" contient plusieurs écritures équiprobables dont certaines intègrent des classes de graphèmes fort éloignées du dessin de la lettre "k". Les autres lettres sont regroupées dans l'annexe A.

fois où un état est inclus dans la meilleure séquence sélectionnée par l'algorithme de Viterbi.

Une sélection pertinente de l'architecture d'un modèle reste pour le moment encore un problème non résolu de manière satisfaisante. Elle aurait pourtant des répercussions sur un possible apprentissage d'une segmentation graphème (voir paragraphes 3.6.6, 4.7) ou sur la pertinence de la reconnaissance sans dictionnaire (voir paragraphe 4.8), ce problème étant relié à une meilleure prise de décision (voir paragraphe 5.3.2).

4.10 Conclusion

Ce chapitre présente la partie de la reconnaissance de l'écriture qui s'étend de la description sous forme de caractéristiques de chaque graphème à l'obtention de la probabilité que l'image contienne tel ou tel mot. Le paragraphe 4.3 a montré qu'un même algorithme peut obtenir des performances médiocres ou bonnes selon le choix de la description des images. Cela tend à montrer que le choix des caractéristiques est de même importance que le choix de la modélisation.

Cette partie développe une méthode de sélection des caractéristiques associées aux graphèmes ainsi qu'une étude de l'association positive d'une séquence de graphèmes et d'une séquence de liaisons. Outre le fait que ce chapitre détaille les mécanismes de la reconnaissance ainsi que l'apport des liaisons dans la reconnaissance, la modélisation markovienne de couples de lettres fréquemment mal segmentées est la contribution majeure présentée dans cette partie.

L'intérêt porté aux méthodes d'apprentissage et aux méthodes de sélection de modèles vient du fait qu'il est parfois nécessaire de comprendre comment le résultat de la reconnaissance a été obtenu. Les réseaux de neurones sont reconnus pour être des boîtes noires qu'il est impossible de déchiffrer. Dans le cas de la reconnaissance, il nous suffit de pouvoir retrouver l'association entre les graphèmes et les modèles de lettres, mais les piètres performances en reconnaissance sans dictionnaire montrent que ce décryptage n'est pas encore tout-à-fait possible bien que les modèles de groupes de lettres soient un pas dans cette direction.

Chapitre 5

Décision

La décision est la dernière étape du processus de reconnaissance et consiste à valider ou invalider les résultats obtenus par les étapes précédentes. Actuellement, les modèles de Markov cachés sont moins fiables que des opérateurs humains et n'atteignent les mêmes performances de reconnaissance que pour la partie des documents la mieux écrite. Par exemple, sur des chèques français, un opérateur humain est capable d'en déchiffrer le montant avec 1% d'erreur, c'est-à-dire que sur cent chèques, un seul n'est pas lu correctement. Ce taux d'erreur est inatteignable par des logiciels de reconnaissance à moins de ne traiter que environ 70% des chèques les mieux écrits, les 30% restant étant toujours décryptés par des opérateurs humains. Par conséquent, l'étape de décision consiste à déterminer l'ensemble des documents pour lesquels les résultats de la reconnaissance peuvent être considérés comme fiables, c'est-à-dire reconnus avec un faible taux d'erreur.

5.1 Courbe taux de lecture substitution / taux d'erreur

Cette courbe permet d'évaluer de manière simple les performances d'un jeu de modèles de reconnaissance qui retourne une réponse (un mot par exemple) et un nombre réel (un critère de confiance). La décision revient à accepter ou rejeter la réponse fournie par le processus de reconnaissance. Ces deux informations permettent d'obtenir pour un seuil fixé S du critère de confiance et pour une base de données quatre nombres :

1. $A_V(s)$: le nombre des documents pour lesquels $C(d) \geqslant s$ et la réponse est bonne.
2. $A_F(s)$: le nombre des documents pour lesquels $C(d) \geqslant s$ et la réponse est mauvaise.
3. $R_F(s)$: le nombre des documents pour lesquels $C(d) < s$ et la réponse est bonne.
4. $R_V(s)$: le nombre des documents pour lesquels $C(d) < s$ et la réponse est mauvaise.

L'objectif de la reconnaissance est de rendre le nombre $A_F(s)$ aussi petit que possible et $A_V(s)$ aussi grand que possible. En faisant varier s, il est possible de tracer le graphe $A_V(s)$ / $A_F(s)$ (voir figure 5.1). Dans la mesure où rejeter par erreur est souvent moins grave que d'accepter par erreur, cette courbe permet, pour un taux d'erreur donné, de déterminer le taux de bonnes acceptations correspondant.

En règle générale, le taux de référence cherché est donné par $\frac{A_V(s)+A_F(s)}{A_V(s)+A_F(s)+R_V(s)+R_F(s)}$ lorsque s est choisi de telle sorte que $\frac{A_F(s)}{A_F(s)+A_V(s)} \leqslant 1\%$. Ce test constitue un moyen simple d'évaluer la pertinence de modèles de reconnaissance. Il suppose évidemment que le critère de confiance ne soit pas une valeur binaire et permette ainsi d'éliminer les cas pour lesquels la reconnaissance n'est pas fiable.

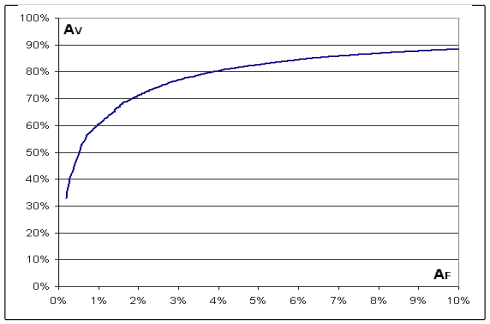


Fig. 5.1: Courbe proche d'une courbe ROC (Receiver Operating Characteristic) : seule la partie correspondant aux faibles taux d'erreur est intéressante. Dans cet exemple, pour 1% de documents reconnus par erreur, 70% le sont correctement.

Ce test est parfois dupliqué lorsque la décision à prendre n'est plus binaire. Dans le cas d'une recherche de mots-clé, il s'agit de déterminer si un mot appartient à un dictionnaire (ou liste des mots-clé) et s'il en fait partie, de dire lequel. Cette expérience est donc constituée de deux décisions :

1. Le mot est-il un mot-clé ? (ou fait-il partie du dictionnaire ?)
2. S'il en fait partie, le mot reconnu est-il le bon ?

Un critère est développé pour chacun des deux problèmes, ou pour les deux simultanément, auxquels sont associés deux seuils de décision. Il n'est pas évident qu'un seul critère soit adapté à ces deux tâches. Toutefois, dans les deux cas, la courbe décrite par la figure 5.1 est toujours utilisée pour évaluer les performances d'un système. Même dans le cas d'un système de décision plus élaboré, cette figure est encore utilisée puisque la règle de décision n'est pas modifiée (critère supérieur ou non à un seuil), la pertinence du système est concentrée dans l'élaboration d'un critère plus "intelligent" qu'une simple probabilité déduite des résultats de la reconnaissance.

Ces courbes [1] permettent de comparer les performances d'un jeu de modèles dans différents contextes où différents jeux de modèles dans les mêmes conditions. Il est encore impossible de concevoir un système de reconnaissance ayant de bonnes performances sur un large éventail de problèmes mais il est possible d'obtenir des performances acceptables pour une tâche particulière. C'est pourquoi les performances sont très sensibles à :

1. La qualité des images en entrée du système : un fond d'image propre, une écriture lisible, une bonne définition de l'image, ...
2. La complexité du résultat : reconnaître un mot inclus dans un petit dictionnaire, reconnaître si un mot est inclus dans une liste, ...
3. Le contexte : la langue par exemple, les écritures anglaise et française sont différentes.

Ces conditions sont sensiblement identiques pour un même problème mais diffèrent souvent pour des documents provenant de sources différentes (scanner, type de documents, type d'information à reconnaître...). Ceci explique pourquoi il est très difficile de comparer différents systèmes de reconnaissance entre eux puisqu'ils sont également estimés sur des bases de données différentes. Cette partie propose trois directions d'étude. La première consiste en une optimisation du processus de reconnaissance, la seconde évoque la fabrication d'un critère plus pertinent en utilisant le maximum d'information retournée par les modèles de reconnaissance, la troisième direction met en correspondance les résultats de différents jeux de modèles de reconnaissance.

1. Annexes : voir paragraphe M, page 422

5.2 Reconnaissance avec dictionnaire, optimisation en vitesse

5.2.1 Introduction

Cette idée est extraite de l'article [Dupré2003] et a pour objectif d'accélérer la reconnaissance d'un document. Lors de la reconnaissance d'un mot avec dictionnaire, il est nécessaire de calculer la probabilité d'une séquence d'observations O pour chaque modèle de mot du dictionnaire D afin de déterminer celui qui obtient le meilleur score :

$$M^* = \underset{M \in D}{\arg\max}\, \mathbb{P}(O \mid M) \tag{5.1}$$

De manière évidente, le temps de reconnaissance augmente linéairement avec la taille du dictionnaire. Nous avons vu au paragraphe 4.8 qu'il est possible d'obtenir la séquence de modèles de lettre la plus probable notée H^*. A cette séquence correspond un mot noté l^*. Ce dernier est rarement un mot correct au sens de la langue et ne peut être le résultat de la reconnaissance mais cette séquence est souvent proche des mots inclus dans le dictionnaire. La proximité de deux mots est dans ce cas retournée par la distance d'édition de Levenstein [2]. On construit donc le voisinage $N^*(s)$ de la séquence de lettres l^* inclus dans un dictionnaire D :

$$N^*(s) = \{m \in D \mid d(m, l^*) \leqslant s\} \tag{5.2}$$

L'idée sous-jacente est de vérifier que le mot M^* défini en (5.1) vérifie également (5.3) :

$$M^* = \underset{M \in N^*(s)}{\arg\max}\, \mathbb{P}(O \mid M) \tag{5.3}$$

L'ensemble $N^*(s)$ doit être suffisamment petit pour éviter un trop grand nombre de calculs des probabilités $\mathbb{P}(O \mid M)$ où M est un modèle de mot, il doit également être suffisamment grand afin d'être sûr que le modèle M^* y soit inclus. Ce compromis se traduit par le choix d'une valeur de s adéquat. Par conséquent, l'optimisation en vitesse repose sur trois étapes :

1. Calculer la séquence de lettres la plus probable en utilisant l'algorithme de Viterbi.
2. Déterminer l'ensemble $N^*(s)$.
3. Calculer pour chaque mot de l'ensemble $N^*(s)$ la probabilité $\mathbb{P}(O \mid M)$ et retourner le mot M^* vérifiant (5.3).

Ce système d'optimisation est résumé par la figure 5.2. Le choix s est un compromis entre optimisation en vitesse et perte de performance.

5.2.2 Résultats

L'obtention de la meilleure séquence de lettres [3] ainsi que le calcul des probabilités ont déjà été exposés au chapitre précédent. L'obtention du voisinage $N^*(s)$ est en fait un problème classique de recherche des plus proches voisins inclus dans un ensemble fini. L'annexe K recense différentes solutions permettant

2. Annexes : voir paragraphe J, page 382
3. Celle-ci est obtenue en utilisant des bi-grammes.

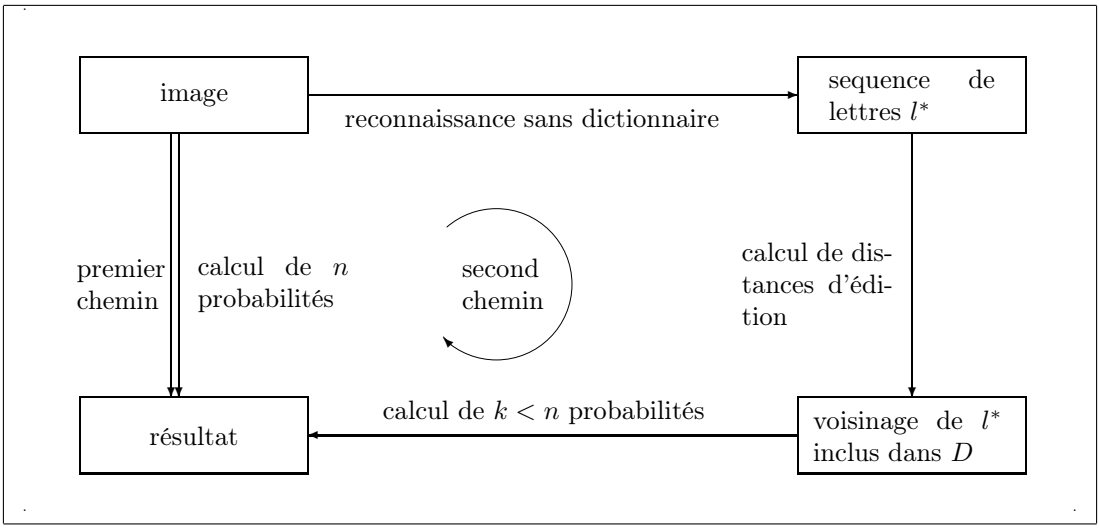


Fig. 5.2: Optimisation en vitesse de la reconnaissance pour un dictionnaire de n mots. Le second chemin est plus rapide que le premier mais est légèrement moins performant.

d'atteindre cet objectif, ces méthodes s'appliquent à tout espace métrique, et donc également à celui des mots muni d'une distance d'édition [4]. L'algorithme LAESA [5] propose une solution simple à mettre en œuvre ne nécessitant pas un prétraitement du dictionnaire trop prohibitif. Les résultats présentés ci-dessous ont été obtenus avec un arbre de partitionnement [6].

Les modèles de reconnaissance utilisés pour mesurer l'amélioration apportée par cette optimisation en vitesse ont été estimés sur une base de 38000 prénoms français et sont testés sur une autre base de 12000 prénoms. Le tableau 5.1 donne les performances et les temps de traitements sans optimisation et pour deux dictionnaires de tailles différentes. Le temps de traitement est une fonction affine de la taille s du dictionnaire de type $\alpha s + \beta$ avec $\beta > 0$ car certains calculs sont communs à chaque mot et factorisés (voir [Koerich2002a], algorithme 4.7.1).

dictionnaire taille	taux de reconnaissance	taux de lecture pour 1% de substitution	temps de traitement
2100	91,9 %	72,9 %	101 ms
11000	83,8 %	46,9 %	244 ms

Tab. 5.1: Performances du système de reconnaissance. Le temps de traitement n'est pas proportionnel par rapport à la taille du dictionnaire parce que certains calculs sont factorisés d'un mot à l'autre (voir [Koerich2002a], algorithme 4.7.1). Ce temps vérifie : $t \sim 0,015\,n + 70\ ms$ où t est le temps de traitement moyen par mot et n la taille du dictionnaire. Ces mesures ont été obtenues sur un Pentium IV 1 GHz.

La seconde expérience a pour objectif de montrer l'évolution des taux et temps de la table 5.1 pour un système optimisé. Le temps de traitement dévolu à chaque document peut être décomposé comme suit :

$$t_{doc} \sim \underbrace{C_{dico} + N_{nn}(s)\ t_{word}}_{\text{temps de reconnaissance}} + \underbrace{C_{viterbi} + C_{nn} + N_{dist}(s)\ t_{dist}}_{\text{temps d'optimisation}} \tag{5.4}$$

4. Annexes : voir paragraphe J, page 382
5. Annexes : voir paragraphe K.3.1, page 409
6. Annexes : voir paragraphe K.1.1, page 391

paramètre	*sens*	*valeur estimée*
C_{dico}	constante de reconnaissance, voir la table 5.1	70 ms
$N_{nn}(s)$	taille moyenne du voisinage, fonction de s	
t_{word}	temps moyen de reconnaissance par mot du dictionnaire	0,015 ms
$C_{viterbi}$	temps moyen pour l'algorithme de Viterbi utilisé pour trouver la séquence de lettres la plus probable	0,01 ms
C_{nn}	constante de la recherche de voisinage	1,8 ms
$N_{dist}(s)$	nombre moyen de distances d'édition calculées, fonction de s	
t_{dist}	temps moyen pour calculer une distance d'édition	0,004 ms

L'expression (5.3) ne permet pas de déterminer le seuil s optimal pour la taille du voisinage car il dépend du dictionnaire et des modèles de reconnaissance. Le tableau 5.2 montre que pour l'expérience décrite précédemment et pour un dictionnaire de 2178 prénoms. Un seuil acceptable est 4, le temps de traitement est alors réduit de 8 millisecondes pour une perte négligeable de performance.

. s	taux de reconnaissance	taux de lecture pour 1% de substitution	taille moyenne moyenne de voisinage $N_{nn}(s)$	nombre moyen de calculs de distances d'édition $N_{dist}(s)$	temps (ms) t_{doc}
0	38,0 %	-	0,4	281	83
1	59,6 %	53,8 %	2,0	508	84
2	78,3 %	65,8 %	11,0	836	86
3	87,1 %	70,8 %	63,3	1227	88
4	**90,6** %	**72,3** %	**266,6**	**1587**	**93**
5	91,7 %	72,9 %	707,1	1792	101
6	91,8 %	72,9 %	1241,1	1753	110
7	91,9 %	72,9 %	1724,4	1490	116

Tab. 5.2: Perte de performance et optimisation en vitesse pour plusieurs valeurs de seuil s (voir l'équation (5.3)), pour un dictionnaire de 2178 mots. La meilleure valeur de s semble 4, le système est plus rapide de 8 millisecondes et le taux de lecture perd seulement 0,6 points.

. s	taux de reconnaissance	taux de lecture pour 1% de substitution	taille moyenne moyenne de voisinage $N_{nn}(s)$	nombre moyen de calculs de distances d'édition $N_{dist}(s)$	temps (ms) t_{doc}
2	71,8 %	40,1 %	36	4045	111
3	79,7 %	44,6 %	281	6366	127
4	82,9 %	45,7 %	1305	8398	154

Tab. 5.3: Perte de performance et optimisation en vitesse pour plusieurs valeurs de seuil s (voir l'équation (5.3)), pour un dictionnaire de 11000 mots. La meilleure valeur de s semble 3, le système est alors deux fois plus rapide et un taux de lecture réduit de 1,6 point.

Avec un dictionnaire de 11000 mots, le gain est plus important, la meilleure valeur est 3, le temps de traitement est divisé par deux pour un taux de lecture réduit de 1,6 point. Le gain en vitesse croît avec la taille du dictionnaire, seul le temps de son prétraitement augmente. La principale amélioration d'un tel

système repose sur l'élaboration d'une distance entre mots intégrant les capacités en reconnaissance du dictionnaire. Cette distance doit être petite pour le couple formé par la meilleure séquence hors dictionnaire et le mot le plus probable du dictionnaire, plus grande pour les autres couples de mots. Elle doit également pouvoir être apprise [7] de manière à s'adapter aux erreurs des modèles de reconnaissance, cet apprentissage est une possible direction de recherche.

5.3 Améliorer le rejet avec un seul jeu de modèles

Dans le cas d'une reconnaissance avec dictionnaire, le premier critère de décision est la probabilité du mot le plus probable. Si elle dépasse un certain seuil, ce mot est considéré comme étant la bonne réponse. Toutefois, afin d'améliorer les performances en reconnaissance, est-il possible de fabriquer un meilleur critère à partir de cette probabilité et d'autres informations telles que la distance d'édition entre le mot le plus probable et la séquence de lettres la plus probable ?

Ces problèmes de décision sont présents dans le cas d'un problème où le dictionnaire est fermé, c'est-à-dire que la réponse se trouve nécessairement dans cet ensemble. Ils sont incontournables lorsque le dictionnaire est ouvert, c'est-à-dire lorsque le système de reconnaissance doit d'abord dire si le mot à reconnaître appartient au dictionnaire et ensuite si tel est le cas, à quel mot l'image correspond. Les dictionnaires ouverts seront malgré tout traités comme des dictionnaires fermés, un mot ayant obtenu un mauvais score sera rejeté, peu importe qu'il soit censé ou non appartenir au dictionnaire.

5.3.1 Informations complémentaires

Le paragraphe 5.2 a montré la proximité entre le mot le plus probable hors dictionnaire et le mot le plus probable inclus dans le dictionnaire. L'idée consiste à construire un score plus pertinent prenant en compte ces informations à l'aide d'un réseau de neurones [8] qui devra retourner une valeur proche de un en cas de réponse positive et proche de zéro pour une réponse négative. Les entrées du réseau de neurones sont les suivantes :

1. la probabilité du mot le plus probable du dictionnaire,
2. la probabilité du second mot le plus probable du dictionnaire,
3. la distance d'édition entre le premier mot et le mot le plus probable hors dictionnaire.

La sortie désirée du réseau de neurones est soit nulle lorsque le mot le plus probable du dictionnaire ne correspond pas au mot écrit, et un dans le cas contraire. Ce procédé ne permet d'améliorer le taux de reconnaissance puisqu'il sert à valider ou non la réponse des modèles de reconnaissance mais il a permis d'augmenter le taux de lecture pour 1% de substitution de 60% à 65% sur une base de 12000 prénoms français et un dictionnaire de 2000 prénoms. Cette amélioration décroît lorsque la taille du dictionnaire augmente car dans ce cas, l'écart entre les deux probabilités les plus élévées diminue ainsi que la distance d'édition entre le premier mot et le plus probable hors-dictionnaire.

La distance d'édition utilisée n'est pas très discriminante puisqu'elle est à valeurs entières et excède rarement cinq. En revanche, il serait possible d'apprendre les coefficients de cette distance [9] de sorte que celle-ci soit élevée lorsque le système de reconnaissance se trompe et faible dans le cas contraire. Cette direction rejoint celle évoquée en conclusion du paragraphe 5.2.

7. Annexes : voir paragraphe J.4, page 388
8. Annexes : voir paragraphe C, page 190
9. Annexes : voir paragraphe J.4, page 388

5.3.2 Mot bruit

Cette idée consiste à ramener le problème d'un dictionnaire ouvert à un dictionnaire fermé en modélisant les mots hors dictionnaire par un ou plusieurs modèles particuliers. Deux versions sont possibles, la première est développée dans [Senior1994] et [Senior1998], elle consiste à ajouter au dictionnaire un mot constitué de toutes les lettres. L'autre version est décrite dans [Augustin2001], elle consiste à inclure dans le dictionnaire plusieurs modèles de bruits dont les coefficients sont estimés sur l'ensemble de la base d'apprentissage.

Pour ces deux versions, le raisonnement est identique. Si le mot à reconnaître n'appartient pas au dictionnaire, il paraît naturel de supposer que la probabilité de ces modèles de mot bruit, capables de tout émettre, sera supérieure à celle de tous les autres mots du dictionnaire. Dans ce cas, l'image sera rejetée.

Les gains de performance obtenus par cette méthode dépendent du dictionnaire utilisé. Plus les mots sont proches les uns des autres, plus il est grand, et plus le rejet sera marginal. Néanmoins, cette méthode permet d'obtenir un gain du même ordre de grandeur que la méthode proposée au paragraphe précédent.

L'intérêt de cette méthode repose également sur le fait que les modèles de lettres n'ont pas été estimés à l'aide de séquences trop bruitées ou que leur architecture a été sélectionnée de manière à réduire l'importance de ces séquences de bruit dans cette estimation. Cette remarque est à rapprocher des conclusions du paragraphe 4.9 concernant la sélection de l'archictecture des modèles de reconnaissance, elles expliquent pourquoi l'introduction d'un modèle de mot bruit pour aider la décision est contrebalancée par la grande variabilité des écritures apprises par les modèles de lettre.

5.4 Mise en parallèle de plusieurs modèles

Jusqu'à présent, le processus d'aide à la décision n'a concerné qu'un seul jeu de modèles de reconnaissance, n'incluant qu'une seule segmentation graphème, qu'un seul jeu de caractéristiques, qu'un seul jeu de modèles de Markov cachés. L'optimisation en vitesse présentée au paragraphe 5.2 permet de réduire le temps de traitement, temps qui pourrait être mis à profit pour comparer les résultats de plusieurs modèles de reconnaissance. L'accroissement de la puissance des ordinateurs permet également de mettre en œuvre un tel système car les temps d'apprentissage sont réduits.

L'article [Wunsch1995] étudie l'apport de plusieurs processus de reconnaissance, la réponse est validée uniquement si tous les résultats convergent vers une unique solution. L'article [Lin2003] propose d'organiser un vote pour trois classifieurs ou plus, le mot qui est le plus souvent reconnu est choisi pour être la solution. Cet article propose une étude théorique à partir de données simulées qui montre l'apport non négligeable de la multiplication des classifieurs. L'article [Günter2004] étudie l'apport de plusieurs classifieurs par rapport à l'amélioration de l'apprentissage d'un classifieur par des méthodes comme *Adaboost*[10] ou la méthode *Bagging*[11]. L'approche proposée par cet article est un système de vote. Toutefois, les classifieurs n'ont pas la même importance, le poids accordé au classifieur n dépend du résultat obtenu par les $n-1$ précédents.

L'expérience suivante utilise des bases d'apprentissage et de test de 7000 mots anglais inclus dans un dictionnaire de 116 mots. La reconnaissance est effectuée par un système de plus proches voisins comme celui décrit au paragraphe 4.4 pour différents jeux de caractéristiques tirés de la table 4.4. Le tableau 5.4 donne les performances obtenues pour chacune d'entre eux. La table 5.5 donne les résultats obtenus par vote. Elle montre que l'utilisation de plusieurs classifieurs permet d'obtenir un taux de reconnaissance meilleur que celui obtenu par le meilleur classifieur. Le vote permet de restreindre la reconnaissance à

10. La méthode Adaboost consiste à pondérer plus fortement les exemples d'apprentissage mal classifiés.

11. La méthode Bagging consiste à apprendre plusieurs classifieurs à partir d'un même échantillon de données incluant N observations. Chaque classifieur est appris sur un ensemble de N observations choisies aléatoirement dans l'échantillon initial. Les points aberrants seront de cette manière sous-représentés, ayant peu de chance d'être choisis dans chaque échantillon.

l'ensemble des documents pour lesquels les résultats sont plus fiables.

	jeu 1	jeu 2	jeu 3	jeu 4	jeu 5	jeu 6
jeux de caractéristiques	$Prof\,(5,5)$	$Mat\,(5,5)$	$Pol\,(30)$	$Ond\,(30)$	$Mat\,(3,3)$	$Som\,(5,5)$
taux de reconnaissance	32,94 %	33,9 %	38,3 %	24,4 %	33,5 %	32,5%

Tab. 5.4: Résultats obtenus sur un même problème par trois processus de reconnaissance différents. Ces caractéristiques sont décrites par la table 4.4 (page 96).

	taux de documents acceptés	taux de reconnaissance
critère maximal	100,0%	39,8%
au moins 2 réponses communes	92,1%	42,7%
au moins 3 réponses communes	60,8%	56,4%
au moins 4 réponses communes	38,8%	70,6%
au moins 5 réponses communes	24,9%	82,6%
au moins 6 réponses communes	13,3%	88,2%

Tab. 5.5: Pour chaque document, la réponse choisie est celle qui est d'abord validée par le plus de reconnaisseurs, puis en cas d'ex aequo, le reconnaisseur ayant le plus fort critère de confiance. Ces critères sont comparables car tous les reconnaisseurs sont des classifieurs utilisant les plus proches voisins.

Le vote est une méthode assez simple permettant d'améliorer les performances en reconnaissance. La suite logique de cette méthode consiste à construire un critère de confiance bâti autour de ceux retournés par les différents classifieurs en utilisant un réseau de neurones [12] par exemple.

Le filtre opéré par la méthode d'optimisation en vitesse présentée au paragraphe 5.2 permet d'envisager plusieurs classifieurs. Ceux-ci ne seraient cependant pas utilisés de manière indépendante les uns des autres car le temps de traitement de chacun est trop coûteux. Chaque reconnaisseur serait utilisé afin d'infirmer ou de confirmer les résultats extraits du précédent reconnaisseur. Par conséquent, chaque reconnaisseur aurait pour objectif de filtrer le dictionnaire afin que le reconnaisseur suivant ne soit utilisé que sur un ensemble resteint de solutions. Ce système de filtres de reconnaissance est une des directions de recherche envisagées.

5.5 Conclusion

L'étape de décision ne modifie pas les taux de reconnaissance mais permet d'améliorer la confiance en ces résultats exprimée par le biais d'un taux de lecture pour un taux donné [13] de substitution. La première méthode présentée tente de conserver ce même taux de confiance tout en améliorant la vitesse de reconnaissance. Ce temps gagné est ensuite réinvesti dans des méthodes permettant d'améliorer cette confiance, soit au niveau du même jeu de modèles, soit en faisant intervenir d'autres reconnaisseurs.

Toutes les méthodes présentées dans ce chapitre sont indépendantes du contexte hormis le fait que la reconnaissance soit restreinte à des mots faisant partie d'un dictionnaire. La décision est toutefois plus facile lorsque le contexte contient des informations supplémentaires permettant d'éliminer des solutions aberrantes. C'est le cas de la reconnaissance d'un chèque où montants littéral et numérique doivent coïncider,

12. Annexes : voir paragraphe C, page 190
13. généralement 1%

le résultat ne peut être que l'ensemble des solutions communes aux deux processus de reconnaissances. Le contexte est une source d'information qu'il ne faut pas négliger lors de la décision.

Chapitre 6

Conclusion et perspectives

Ce document a présenté l'ensemble du processus de reconnaissance d'un mot manuscrit, depuis l'image jusqu'au résultat final constitué d'une réponse et d'un critère de confiance. La première direction de recherche envisagée a été l'amélioration des modèles de reconnaissance mais au fur et à mesure des articles lus et des expériences realisées, les limites de cette voie sont apparues. Les méthodes de sélection des coefficients des modèles présentées dans ces travaux n'ont pas de véritable impact sur les performances. Toutefois, si cette voie n'a pas permis d'établir une méthode efficace de construction de l'architecture des modèles de reconnaissance associés aux lettres, elle a débouché sur un nouvel agencement des modèles de mot permettant d'inclure des modèles de reconnaissance associés aux erreurs fréquemment produites par le traitement d'images.

Plutôt que de chercher à améliorer les performances des modèles de reconnaissance, ces travaux ont ensuite été orientés vers les traitements d'images et notamment la segmentation en graphèmes. L'objectif était dans un premier temps de fabriquer une segmentation délivrée des habituelles heuristiques introduites dans le traitement d'image, c'est-à-dire une segmentation capable d'être apprise à partir d'images présegmentées avec, à plus long terme, l'intention d'appliquer cette méthode à la segmentation d'une image en lettres. Les premiers résultats obtenus sur un problème classique de segmentation en lignes ont été décevants et cette piste a été abandonnée au profit d'une segmentation en graphèmes à nouveau basée sur des heuristiques mais traitant les accents de manière séparée, améliorant légèrement les performances pour une langue incluant ces particules aériennes.

Les systèmes de reconnaissance de mots cursifs utilisent souvent un seul modèle de reconnaissance alors que la reconnaissance de caractères agrège les résultats de plusieurs classifieurs qui améliorent les performances de chacun pris séparément. Par extension, un système de reconnaissance ne devrait pas fonder sa décision sur un seul jeu de modèles de lettre. Il apparaît donc préférable de varier les traitements plutôt que de trop chercher à les perfectionner. Ce constat a orienté les recherches vers une méthode permettant d'accroître la vitesse du système de décision en restreignant la recherche d'une solution à une partie seulement du dictionnaire. Le temps ainsi gagné encourage l'utilisation simultanée de plusieurs jeux de modèles de reconnaissance.

La suite pressentie à ces travaux s'articule autour de trois axes principaux. Le premier consiste à utiliser le temps gagné par l'optimisation en vitesse du processus de décision afin de filtrer successivement les résultats de la reconnaissance par plusieurs jeux de modèles de Markov cachés. Tout d'abord, un premier reconnaisseur "dégrossit" le problème, le second reconnaisseur et les suivants effectuent une reconnaissance plus poussée sur un ensemble de solutions réduit par le filtre précédent. C'est une perspective de recherche centrée sur des méthodes de décision.

Le second axe de recherche concerne l'interprétation des modèles de reconnaissance ou leur estimation de manière à ce qu'ils soient plus interprétables. Pour le moment, la reconnaissance de l'écriture est contrainte à l'utilisation d'un dictionnaire. En résumé, les reconnaisseurs savent comparer deux solutions et déterminer la meilleure mais ne peuvent pas encore aboutir à cette solution sans a priori, ce qui caractérise la reconnaissance sans dictionnaire. La création de modèles de groupe de lettres a été envisagée dans le but de modéliser les défauts de la segmentation en graphèmes. Toutefois, même définis ainsi, ces modèles agrègent trop de bruit lors de l'apprentissage, en particulier pour les lettres insuffisamment représentées. Les premières réponses envisagées sont la suppression des images aberrantes ou la multiplication des images peu fréquentes. Le but visé reste malgré tout une méthode pertinente de sélection de la structure des modèles.

La dernière perspective concerne l'apprentissage d'une segmentation. Cette étape ne permet pas encore d'ajustements autre que manuel, les objets sont scindés selon des heuristiques, des seuils, qui ne sont pas dérivables. Il n'existe pas encore d'algorithme capable d'optimiser ces seuils. Peu de résultats ont été trouvés dans ce domaine. L'idée développée dans ce document n'a pas abouti à des résultats satisfaisants car elle s'appuyait sur un diagramme de Voronoï. Celui-ci pourrait être remplacé en première approche par des cartes de Kohonen, moins sensibles aux pixels isolés. Cette direction demeure attrayante.

Ce manuscrit décrit tous les algorithmes entrant dans la conception d'un système de reconnaissance de mots isolés. La plupart de ces méthodes sont indépendantes du problème à résoudre. Les trois étapes fondamentales sont la segmentation en graphèmes, la reconnaissance à l'aide de modèles probabilistes et la décision. Ce sont généralement les trois briques communes à tout système de reconnaissance de l'écriture cursive. Ce rappel succinct met en relief la multitude d'algorithmes qui participe à l'élaboration d'un logiciel de reconnaissance et pour laquelle il est difficile de contourner une première phase de réflexion autour d'une architecture logicielle. Celle-ci doit être capable notamment d'intégrer à la fois la reconnaissance et l'apprentissage qui nécessitent la manipulation de grandes quantités d'information. L'exigence de rapidité implique souvent une chasse constante de la redondance dans les programmes informatiques. Ainsi, les différentes contributions de ces travaux de thèse, souvent implémentées de manière séparée, sont sur le point de trouver un refuge commun au sein d'une même architecture.

Chapitre 7

Nouveaux enjeux

A partir de la reconnaissance d'un mot isolé cursif, il est possible de décrypter des documents assez fortement structurés où figurent un nom, un prénom, un numéro d'identification, ..., chacun placé à une position variant très peu d'un document à l'autre. C'est le cas d'une feuille de maladie, d'un formulaire quelconque. On peut considérer que ces problèmes sont en partie résolus puisque les performances obtenues par les logiciels de reconnaissance sont plus intéressante que des solutions manuelles pour les entreprises qui les utilisent. Les améliorations à venir consistent presque seulement à faire progresser les taux de reconnaissance, les temps de traitement, bénéfice en grande partie assuré par l'accroissement constant des performances des ordinateurs.

Toutefois, ces performances, quoique suffisantes prises isolément, ne permettent pas de reconnaître l'ensemble des mots d'un paragraphe. Avec un taux de reconnaissance d'un mot cursif avoisinant les 90%, il suffit d'une phrase de sept mots pour que ce taux tombre en dessous de 50% ($0,9^7 \sim 0,47$). Il paraît donc inconcevable de lire un paragraphe sans erreur.

Deux directions de recherche émergent. La première consiste à extraire certaines informations plus importantes à l'intérieur d'un paragraphe comme la présence d'un ou plusieurs mot-clés, en particulier la présence d'un identifiant, un numéro de sécurité sociale, une référence propre à une entreprise, un numéro de téléphone qui est un des sujets du paragraphe 7.1. La seconde direction s'intéresse aux informations qui pourraient aider au décryptage complet d'un paragraphe comme la modélisation du langage associé au paragraphe (voir paragraphe 7.2.3). Par exemple, une ordonnance rédigée par un médecin n'utilise ni le même vocabulaire, ni la même grammaire qu'un jugement rendu par un tribunal.

7.1 Extraction d'informations ciblées à l'intérieur d'un paragraphe

7.1.1 Détection de chiffres dans une lettre manuscrite

On cherche ici à extraire des informations exclusivement numériques insérées dans un texte entièrement manuscrit telle qu'un code postal, un numéro de téléphone, un numéro de client (voir figure 7.1). La méthode décrite ici est celle de l'article [Koch2005]. Les techniques utilisées sont effectivement celles de la reconnaissance de l'écriture manuscrite, toutefois, les auteurs constatent qu'il est impossible de reconnaître l'ensemble des mots du document afin d'en extraire les informations numériques : le temps de traitement est trop important [1].

1. Ce temps varie selon la masse de texte à reconnaître, il peut atteindre quelques minutes pour les documents comme ceux de la figure 7.1. Lorsqu'une information numérique est extraite, le courrier peut alors être dirigé automatiquement vers

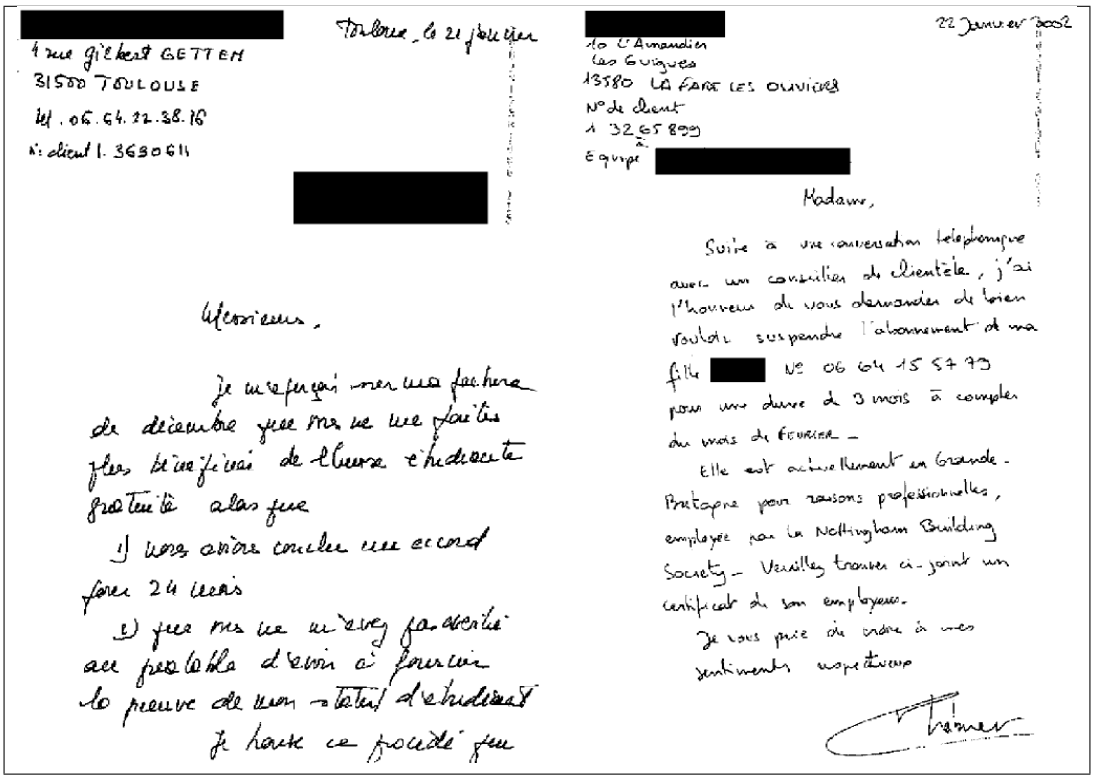


Fig. 7.1: Exemple de documents contenant des informatiques numériques telles qu'un code postal, une date, un numéro de téléphone, un numéro de client, figure extraite de [Koch2005].

La méthode que les auteurs proposent pour extraire les informations numériques se décomposent en quatre étapes :

1. Le document est segmenté en lignes, ils s'inspirent pour cela de l'article [Likforman1995] (voir également le paragraphe 3.3.2).
2. Sur chaque ligne sont extraites les composantes connexes.
3. Chaque composante connexe est classée selon quatre catégories :
   (a) rejet (R) : ce n'est pas un chiffre
   (b) chiffre (D)
   (c) séparateur (P) : virgule, point, barre pour une date
   (d) chiffre double (DD) : une composante connexe représente deux chiffres
4. Une fois chaque composante connexe classée, on effectue en quelque sorte la recherche d'une expression régulière associée soit à un code postal, à une date, à un numéro de téléphone. Cette expresssion régulière est modélisée par une chaîne de Markov.

La première étape ne sera pas plus décrite ici, l'article ne la décrit d'ailleurs pas en détail. La seconde consiste à extraire des composantes connexes. La troisième est une classification effectuée à l'aide de la méthode des plus proches voisins (voir paragraphe I.1). A chaque composante est associée un vecteur de caractéristiques. Celui dépend de trois données :

---

la personne en charge du dossier. Comme toute chaîne de traitement, l'extraction de ces informations ne marche pas sur tous les documents, elle n'est fiable qu'avec un certain taux de confiance. Si en plus, le temps de traitement est rédhibitoire, l'automatisation du tri n'est plus intéressante.

1. H_C est la hauteur de la boîte englobante
2. W_C est la largeur de la boîte englobante
3. G_C est l'abscisse du barycentre de la composante connexe

On suppose que si C désigne une composante connexe, $C-1$ désigne celle qui la précède sur la même ligne, et $C+1$ celle qui la suite. On construit le vecteur $(f_1, \ldots, f_7)$:

$$\begin{array}{lcl|lcl} f_1 & = & \frac{H_{C-1}}{H_C} & f_2 & = & \frac{H_{C+1}}{H_C} \\ f_3 & = & \frac{W_{C-1}}{W_C} & f_4 & = & \frac{W_{C+1}}{W_C} \\ f_5 & = & \frac{|G_C - G_{C-1}|}{W_C} & f_6 & = & \frac{|G_C - G_{C+1}|}{W_C} \\ f_7 & = & \frac{H_C}{W_C} & & & \end{array} \tag{7.1}$$

Un résultat de la classification est représenté par la figure 7.2. On conserve alors pour chaque composante la meilleure classe. Mise bout à bout, on obtient une séquence de classes aussi longue qu'il y a de composantes connexes.

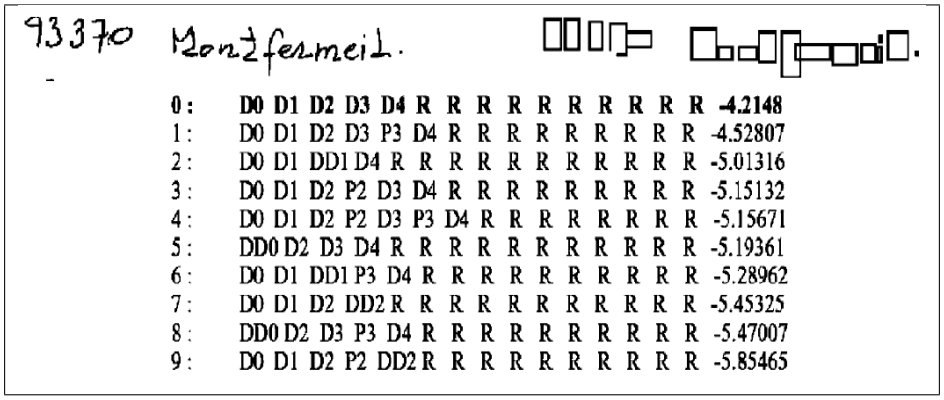


Fig. 7.2: Classification des composantes en utilisant les plus proches voisins et les caractéristiques (7.1), figure extraite de [Koch2005]. Chaque ligne représente une solution possible, le dernier chiffre est le coût de la solution.

On modélise ensuite l'information numérique qu'on veut chercher. La figure 7.3 représente le modèle associé à un code postal. On calcule ensuite la probabilité de chaque sous-séquence de classe connaissant le modèle. Selon les scores obtenus, on décide ou non de valider la présence d'un code postal.

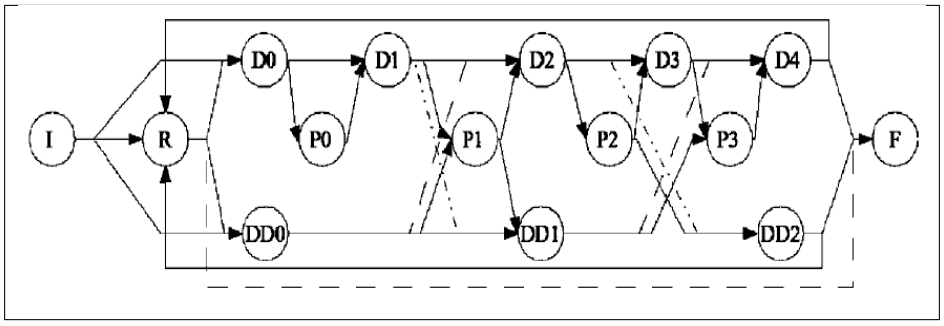


Fig. 7.3: Chaîne de Markov représentant un code postal, figure extraite de [Koch2005]. L'écriture la plus probable est cinq chiffre : D0,D1,D2,D3,D4. On peut également intercaler du bruit (P), et des double chiffres résultant de chiffre se touchant.

7.2 Dématérialisation des flux entrants de documents

Certaines entreprises reçoivent des quantités de courrier incroyablement grande qu'il faut traiter chaque jour. Une banque peut recevoir par exemple jusqu'à 50000 fax correspondant à des ordres de virement qu'il faut exécuter. D'autres sociétés reçoivent des lettres de réclamations dans lesquels un numéro de référence, de téléphone est rappelé. Chaque jour, ce courier doit être traité, pas plusieurs dizaine de personnes qui finissent par se spécialiser dans tel ou tel type de courrier. Chaque courrier est donc classé pour adressé à la personne qui le traitera, à commencer par la saisie d'un numéro de référence. Ces deux thèmes sont abordés dans les annexes [2].

7.2.1 Classification

La classfication d'un document revient à identifier soit un formulaire (une disposition graphique), soit un document dont le texte exprime explicitement l'action à accomplir.

7.2.2 Extraction d'information

Ces informations apparaissent partout sur une page, souvent à côté d'un mot-clé. On écrit souvent son nom, son prénom à côté d'un mot imprimé "nom" ou "prénom". Il s'agit de chercher dans une page ce mot imprimé et de reconnaître l'information présente à droite ou juste à côté dessus ou dessous.

7.2.3 Contexte, modélisation du langage

Ces méthodes, jusque-là réservées au traitement du langage naturel, commencent à être utilisées en reconnaissance de l'écriture. Le décryptage d'une page complète souffre des performances pas encore suffisantes des reconnaisseurs de caractères. Les méthodes de langage permettent d'éviter les solutions qui n'ont pas de sens.

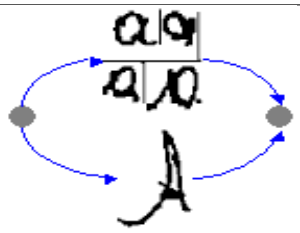
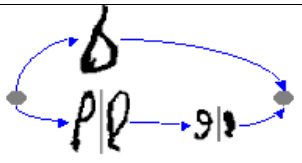
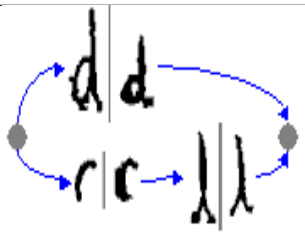
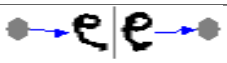
2. Annexes : voir paragraphe N, page 433

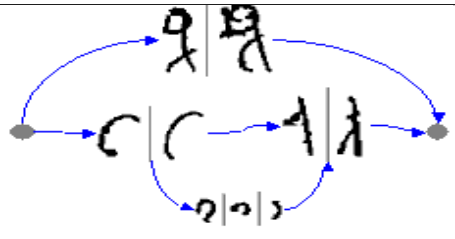
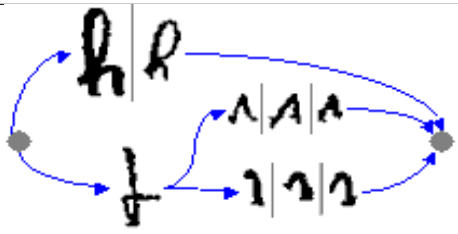
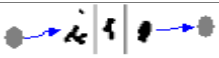
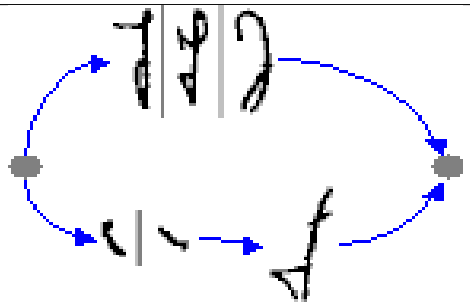
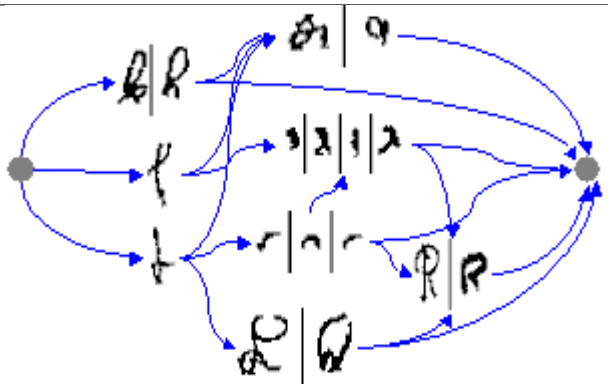
Annexes

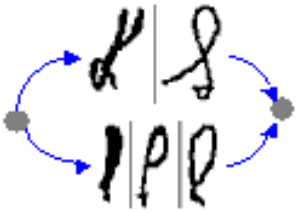
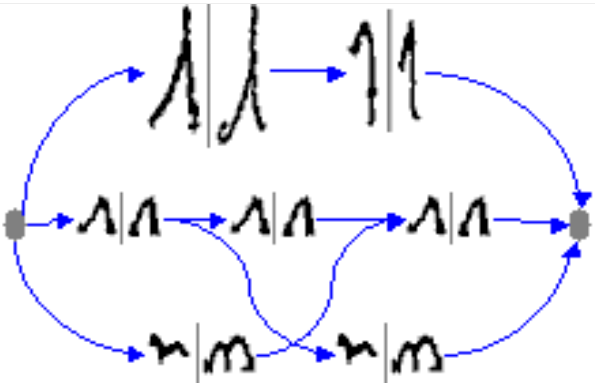
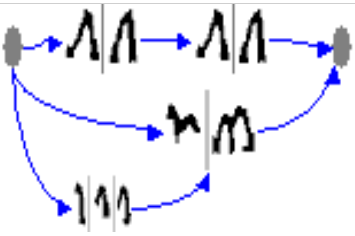
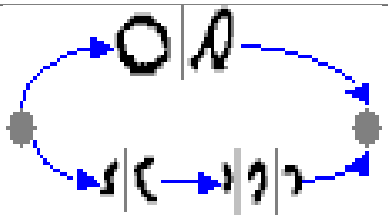
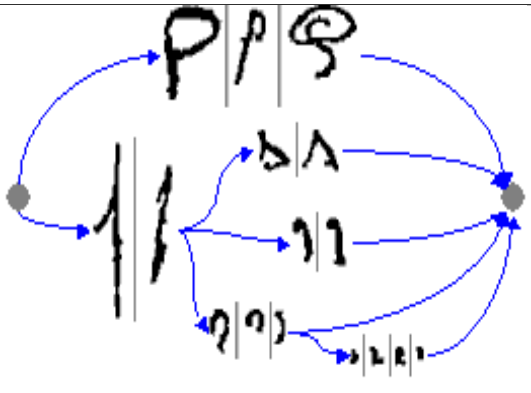
Annexe A

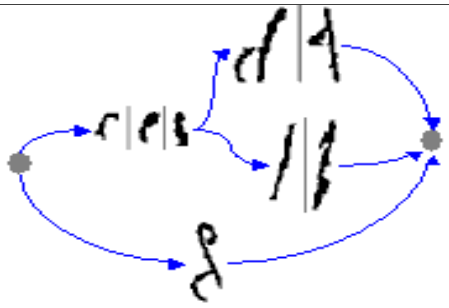
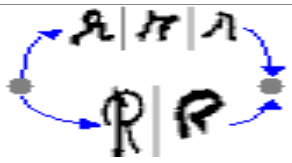
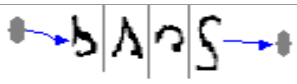
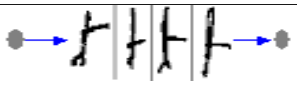
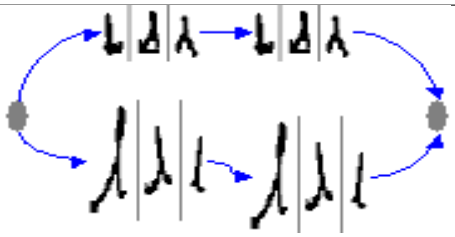
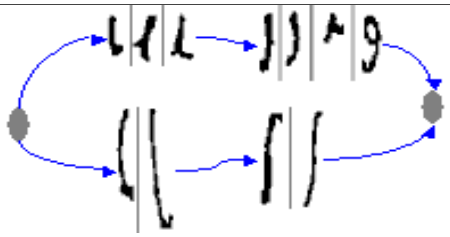
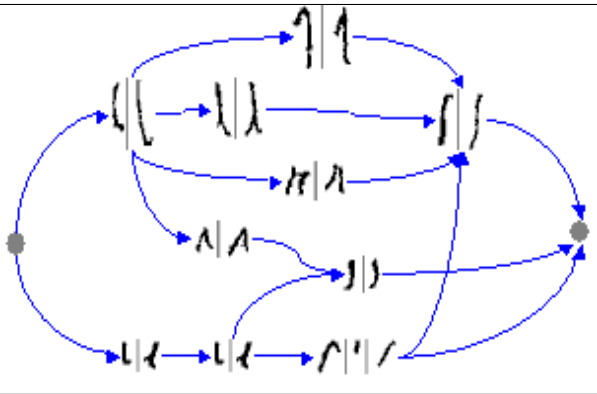
Illustration de modèles de lettres

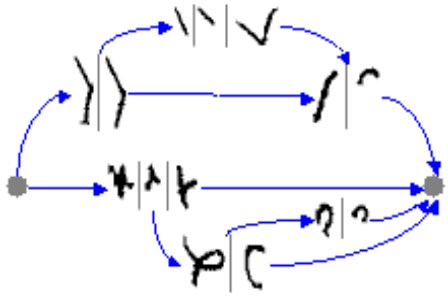
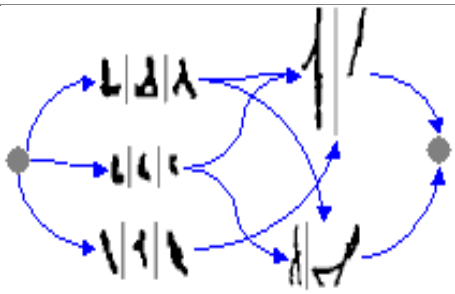
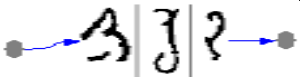
Cette annexe regroupe pour chaque lettre une représentation simplifiée d'un modèle de Markov caché qui lui est associé. Chaque état émet une classe d'observations illustrée par un ou deux de ses éléments les plus probables. Chaque image de modèle résume les écritures les plus fréquentes pour chaque lettre. Ces lettres ont été estimées sur une base d'apprentissage de 30000 prénoms français. Seuls ont été conservées les transitions dont les probabilités sont supérieures à 0,15 afin de ne garder que les principaux coefficients sur les quelques centaines que contient chaque modèle. Leur architecture a été sélectionnée à partir des méthodes développées dans l'annexe F.

lettre	modèle
a	
b	
c	
d	
e	

lettre	modèle
f	
g	
h	
i	
j	
k	

lettre	modèle
l	
m	
n	
o	
p	

lettre	modèle
q	
r	
s	
t	
u	
v	
w	

lettre	modèle
x	
y	
z	

Les lettres les moins fréquentes "k", "w", "x" ont des modèles incluant des symboles parfois éloignés du dessin de la lettre. Lors de leur apprentissage, étant donné le faible nombre d'exemples - seulement une centaine pour la lettre "w" - servant à l'estimation de leur coefficients, chaque image représente une écriture chaque fois presque unique. Il suffit alors qu'une erreur de segmentation intervienne ou que le mot soit mal écrit pour que le modèle inclut une écriture peu vraisemblable de même importance que les autres. C'est le cas des lettres "y" ou "x" pour lesquels on ne dispose que peu d'exemples d'apprentissage. En revanche, la lettre "z" pourrait être considérée comme une lettre rare elle aussi mais les prénoms présents dans la base d'apprentissage balayent tout un siècle. Des prénoms comme Elizabeth, Gonzague, Suzanne ne sont pas rares.

Annexe B

Squelettisation

La squelettisation est une opération qui permet de passer d'une image aux traits épais à une représentation en "fil de fer" dont la première apparition date de [Blum1967]. La figure B.1 représente à la fois l'image initiale et le résultat obtenu en filigrane. Cette représentation "fil de fer" est appelée *squelette*. Il n'existe pas de définition unique du squelette, il doit seulement respecter la connexité de la forme qu'il est censé représenter ou plus précisément, une forme et son squelette doivent être *homotopes*. Pour une composante connexe, le squelette correspondant ne forme également qu'une seule composante connexe incluse dans la première.

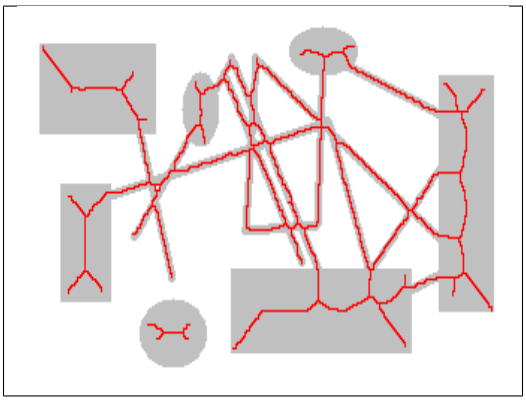


Fig. B.1: Squelettisation d'une figure quelconque

L'article [Lam1992] propose une revue des algorithmes de squelettisation. Ce chapitre présente quelques-unes des méthodes qui y sont présentées comme la squelettisation par érosion (voir paragraphe B.4.1), ainsi que d'autres issues d'articles plus récents et mieux adaptées à la reconnaissance de l'écriture comme la modélisation des intersections (voir paragraphe B.6.4).

B.1 Squelette d'une forme continue

On se place ici dans le plan $\mathbb{R}^2$ et on considère une partie F du plan qu'on appellera par la suite *forme continue*. C'est en général une partie connexe d'intérieur non vide. L'objectif est ici de définir le squelette de cette forme, c'est-à-dire une partie d'intérieur vide la représentant le mieux possible. On définit une boule comme un disque de rayon strictement positif du plan. Les définitions qui suivent restent valables

pour des espaces de dimension supérieure à deux (voir [Blum1973], [Thiel1994]).

Définition B.1.1 : boule maximale
Soit F une forme continue, on note $\mathcal{B}_F$ l'ensemble des boules incluses dans F. Une boule B est dite maximale si et seulement si :

$$B \text{ est maximale } \iff \forall B' \in \mathcal{B}_F,\ B \subset B' \Longrightarrow B = B'$$

Par conséquent, une boule B est maximale pour la forme F si aucune autre boule incluse dans cette forme n'inclut B.

Définition B.1.2 : axe médian
L'axe médian d'une forme F est le lieu des centres des boules maximales de cette forme.

Un exemple d'axe médian est donné par la figure B.2. L'axe médian est aussi un moyen de compresser l'information puisque la description de cet axe ainsi que la donnée du rayon de la boule maximale pour chaque point qui en est le centre permet de reconstituer exactement la forme ([Rosenfeld1986]).

L'axe médian n'est pourtant pas encore le squelette car, même si une forme F n'a qu'une composante connexe, l'axe médian peut en avoir plusieurs. Il suffit, pour s'en convaincre, de considérer deux disques tangents comme l'illustre la figure B.3. Axe médian et forme ne sont donc pas forcément homotopes.

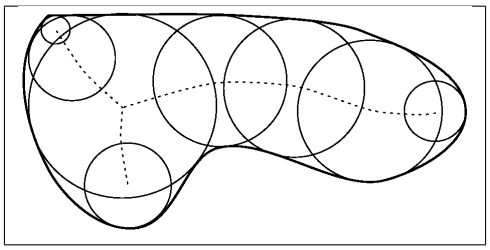


Fig. B.2: Axe médian d'une forme continue (trait pointillé) dont seul le contour est représenté, figure extraite de [Thiel1994].

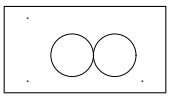


Fig. B.3: Deux disques tangents : cette forme ne contient qu'une seule composante connexe alors que son axe médian n'est constitué que de deux points, les deux centres des disques.

L'axe médian, qui est unique, sert de base à la construction d'un squelette, qui ne l'est pas. Il est possible de prolonger ses extremités afin d'obtenir un ensemble homotope à la forme d'origine. Mais il n'est pas toujours évident de les prolonger et c'est pourquoi il est parfois préférable de rogner la forme jusqu'au squelette, en particulier dans le cas du traitement d'images discrétisées qui est l'objet de ce chapitre.

B.2 4-connexité ou 8-connexité

Afin d'être capable de construire le squelette d'une forme, la notion de connexité doit être transposée du plan à l'ensemble discret de pixels que forme une image. La figure B.4 représente un disque et un carré qui peuvent, selon la définition de la connexité par arc dans une image, être ou ne pas être scindés. Les

deux figures (disque et carré) n'en forment qu'une seule si tout point de l'une peut être relié à tout point de l'autre par un chemin inclus dans la réunion des deux.

Définition B.2.1 : chemin
Un chemin C allant du pixel x au pixel y est une succession de petits déplacements $(v_i)_{1 \leqslant i \leqslant n}$ telle que :

$$y = x + \sum_{i=1}^{n} v_i$$

Ce chemin C est inclus dans un ensemble de pixels P si tous les points empruntés par ce chemin appartiennent à P :

$$C \text{ est inclus dans } P \iff x \in P \text{ et } \forall k \in \{1, \ldots, n\},\ x + \sum_{i=1}^{k} v_i \in P$$

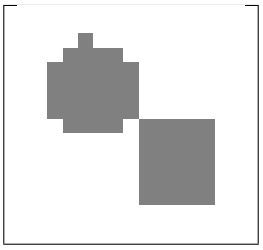


Fig. B.4: Un disque et un carré qui se "touchent" en 8-connexité mais qui sont séparés en 4-connexité.

On définit deux ensembles de déplacements élémentaires, E_4 et E_8 :

$$\begin{aligned} E_4 &= \left\{ \binom{1}{0}, \binom{0}{1}, \binom{-1}{0}, \binom{0}{-1} \right\} \\ E_8 &= E_4 \cup \left\{ \binom{1}{1}, \binom{-1}{1}, \binom{-1}{1}, \binom{-1}{-1} \right\} \end{aligned}$$

Définition B.2.2 : chemin k-connexe
Le chemin $C = (v_i)_{1 \leqslant i \leqslant n}$ est dit k-connexe si $\forall i \in \{1, \ldots, n\},\ v_i \in E_k$.

Ceci nous permet de définir la 4-connexité et la 8-connexité.

Définition B.2.3 : 4-connexité et 8-connexité
On considère une image I et P une partie de cette image. On dit que P est k-connexe ($k \in \{4, 8\}$) si et seulement si pour tout couple de pixels $(x, y) \in P^2$, il existe un chemin k-connexe inclus dans P allant de x à y.

Le code de *Freeman* est très utilisé pour représenter les chemins 4-connexe ou 8-connexe. Il consiste à associer à chaque déplacement de E_8 un numéro en tournant dans le sens inverse des aiguilles d'une montre.

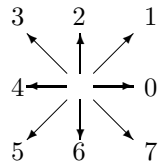
$\binom{1}{0}$	0	$\binom{1}{1}$	1		
$\binom{0}{1}$	2	$\binom{-1}{1}$	3		
$\binom{-1}{0}$	4	$\binom{-1}{-1}$	5		
$\binom{0}{-1}$	6	$\binom{1}{-1}$	7		

Fig. B.5: Code de Freeman permettant de coder les huit directions éléments d'un pixel vers un de ses huit voisins en 8-connexité.

$\begin{matrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \end{matrix}$	$\begin{matrix} \sqrt{2} & 1 & \sqrt{2} \\ 1 & 0 & 1 \\ \sqrt{2} & 1 & \sqrt{2} \end{matrix}$ ou $\begin{matrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{matrix}$	$\begin{matrix} 7 & 6 & 5 & 6 & 7 \\ 6 & 4 & 3 & 4 & 6 \\ 5 & 3 & 0 & 3 & 5 \\ 6 & 4 & 3 & 4 & 6 \\ 7 & 6 & 5 & 6 & 7 \end{matrix}$
Masque de la distance utilisée pour calculer la carte de la figure B.6.	Masques correspondant à la distance euclidienne, le premier est peu utilisé car il implique des calculs réels plus longs que des calculs sur des entiers. $\sqrt{2}$ est de préférence estimé par un rationnel, $\frac{3}{4}$ ou $\frac{5}{7}$.	Exemple de masque 5x5, si ce masque est noté $M \in M_5(\mathbb{R}) = (m_{ij})_{-2 \leqslant i,j \leqslant 2}$, lorsque $m_{20} > 2m_{10}$, le calcul de distance défini en B.3.3 ne fera jamais intervenir le déplacement $\binom{2}{0}$.

Tab. B.1: Masques de distance : chaque case contient la distance d'un pixel au pixel central de la matrice.

B.3 Carte de distance

Définition B.3.1 : carte de distance
Soit I_{XY} une image binaire, avec Y lignes et X colonnes. $I_{XY}(x,y) \in \{0,1\}$ désigne le pixel de coordonnées (x,y). Soit d une distance entre pixels. On désigne par F l'ensemble des pixels noirs de l'image I_{XY}, soit $F = \{(x,y) \mid I_{XY}(x,y) = 1\}$. La carte de distance $C^{I_{XY}}$ est une matrice de mêmes dimensions que l'image vérifiant :

$$\forall (x,y),\ C^{I_{XY}}(x,y) = \min\left\{d\left[(x,y),(x',y')\right] \mid (x',y') \in \overline{F}\right\} \tag{B.1}$$

La figure B.6 illustre une carte de distance pour une distance qui associe à deux pixels le nombre de déplacements verticaux ou horizontaux nécessaires pour aller de l'un à l'autre. Pour cette figure, la valeur d'une case correspond à la distance entre le pixel noir considéré et le pixel blanc le plus proche. Cette distance est introduite car ses maxima locaux ont une forte probabilité d'appartenir au squelette. Il n'existe pas une unique distance (voir [Arcelli1985]), elles sont définies en général par leur masque (table B.1) qui donne leur valeur dans un petit voisinage. Masque et distance sont définis comme suit :

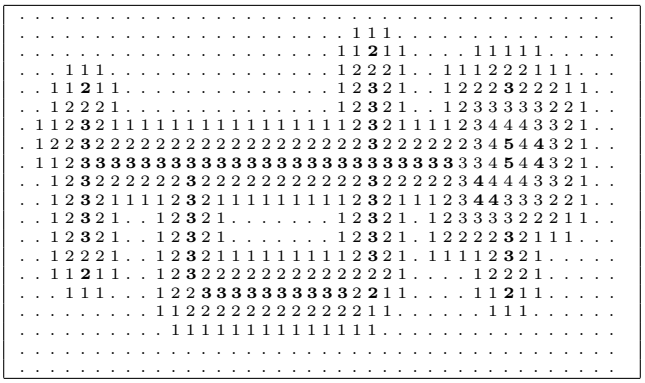


Fig. B.6: Carte de distance : la distance entre deux pixels correspond au nombre de déplacements horizontaux et verticaux nécessaires pour aller de l'un à l'autre. La valeur d'une case correspond à la distance entre le pixel noir considéré et le pixel blanc le plus proche. Les maxima locaux (en gras) ont de forte chance d'appartenir au squelette. Son masque est le premier de la table B.1. Le contour de la forme est l'ensemble des points pour lesquels la carte retourne la valeur 1.

Définition B.3.2 : masque de distance
Soit $M \in M_{2n+1}(\mathbb{R}) = (m_{ij})_{-n \leqslant i,j \leqslant n}$ une matrice carrée de dimension $2n+1$. La matrice M est un masque de distance de dimension n si :
- $\forall (i,j) \neq (0,0),\ m_{ij} > 0$
- $m_{00} = 0$
- M est symétrique selon le pixel central

Définition B.3.3 : distance induite par un masque
Soit M un masque de dimension n, soient deux pixels p_1, p_2 et $C(p_1, p_2)$, l'ensemble des chemins allant de p_1 et p_2 dont les vecteurs sont de longueur inférieure ou égale à n. Alors :

$$d_M(p_1, p_2) = \min_{c \in C(p_1,p_2)} \sum_{v=\binom{v_x}{v_y} \in c} m_{v_x, v_y}$$

L'application définie en B.3.3 est bien une distance. Comme le masque est symétrique, elle est aussi symétrique. De plus, $d_M(p_1, p_2) = 0 \iff p_1 = p_2$. L'inégalité triangulaire est aussi facile à vérifier puisque pour trois points p_1, p_2, p, la concaténation des chemins allant de p_1 à p, puis de p à p_2 forme un chemin allant de p_1 à p_2. Par conséquent : $d_M(p_1, p_2) \leqslant d_M(p_1, p) + d_M(p_2, p)$.

L'algorithme qui suit permet d'obtenir la carte B.6 à partir d'une distance définie en B.3.3 de manière rapide, soit en deux "passes" d'image. Plus précisément, il est nécessaire de parcourir deux fois l'ensemble des pixels de l'image afin de construire la carte de distance. Auparavant, on définit les deux voisinages de pixels suivants :

$$V_h(n) = \left\{ v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \;\middle|\; -n \leqslant v_y \leqslant 0 \text{ et } \begin{cases} -n \leqslant v_x < 0 \text{ si } v_y = 0 \\ -n \leqslant v_x \leqslant n \text{ si } v_y < 0 \end{cases} \right\}$$

$$V_b(n) = \left\{ v = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \;\middle|\; 0 \leqslant v_y \leqslant n \text{ et } \begin{cases} 0 < v_x \leqslant n \text{ si } v_y = 0 \\ -n \leqslant v_x \leqslant n \text{ si } v_y > 0 \end{cases} \right\}$$

Ces deux voisinages sont résumés par la figure B.7.

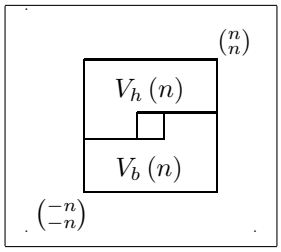


Fig. B.7: Voisinages $V_h(n)$ et $V_b(n)$, le voisinage $V_h(n)$ désigne la partie située au-dessus et à gauche du pixel central, $V_b(n)$ la partie située au-dessous à et à droite.

Algorithme B.3.4 : carte de distance
L'objectif est de construire la carte de distance C pour la forme F et définie par (B.1) pour la distance induite par le masque M de dimension n. I_{XY} est une image avec X colonnes et Y lignes. $\forall (x,y),\ I_{XY}(x,y) = \mathbb{1}_{\{(x,y)\in F\}}$. Pour ne pas alourdir les notations, $I = I_{XY}$ et $C = C^{I_{XY}}$. On impose que $C(x,y) = I(x,y) = \infty$ si $(x,y) \notin \{1,\cdots,X\} \times \{1,\cdots,Y\}$.

Etape A : première passe d'image
pour $y = 1$ **à** Y **faire**
pour $x = 1$ **à** X **faire**

$$C(x,y) \longleftarrow \begin{cases} 0 \text{ si } I(x,y) = 0 \\ \min\left\{C\left[(x,y) - v\right] + M_{v_x,v_y} \mid v \in V_h(n)\right\} \text{ sinon} \end{cases}$$

fin pour
fin pour

Etape B : seconde passe d'image
pour $y = Y$ **à** 1 **faire**
pour $x = X$ **à** 1 **faire**

$$C(x,y) \longleftarrow \min\left\{C(x,y), \min\left\{C\left[(x,y) - v\right] + M_{v_x,v_y} \mid v \in V_b(n)\right\}\right\}$$

fin pour
fin pour

L'algorithme B.3.4 mène bien à la carte de distance définie par (B.1). Cette démonstration est assez simple et s'effectue par récurrence sur x et y. La première passe détermine pour un pixel noir (1) la distance au pixel blanc (0) le plus proche situé dans une moitié supérieure de l'image. La seconde passe détermine ce minimum dans la moitié inférieure et choisit le minimum des deux.

B.4 Squelettisation discrète

La squelettisation doit aboutir à une image style "fil de fer" comme celle de la figure B.1. Il existe plusieurs méthodes de squelettisation discrète, certaines s'appuient sur une carte de distance comme celle de la figure B.6. D'autres approches sont cependant possibles comme par l'intermédiaire d'un graphe de Voronoï (paragraphe B.5.1).

B.4.1 Erosion à partir de masques $(3,3)$

L'érosion d'une forme consiste à lui retirer l'ensemble des pixels qui forme son contour. Toutefois, l'érosion d'une forme composée d'une seule composante connexe peut aboutir à plusieurs composantes. Son squelette est donc obtenu après plusieurs érosions successives tout en conservant à chaque étape des formes homotopes à la forme initiale.

2	1	2		1	1	1		
1	0	1		1	0	1		
2	1	2		1	1	1		
M_4				M_8				

Fig. B.8: Masques de distance M_4 et M_8 utilisés pour construire un squelette en k-connexité.

Tout d'abord, on définit la fonction f_k qui associe à un pixel un ensemble de pixels correspondant à un de ses voisinages définis par la figure B.9 (4 ou 8 connexité). On définit également pour une forme F :

$$f_k(F) = \bigcup_{p \in F} f_k(p)$$

Les algorithmes qui suivent supposent fréquement que la forme à squelettiser ne contient qu'une seule composante connexe. Il suffit de répéter cet algorithme pour chaque composante connexe si ce n'est pas le cas.

Algorithme B.4.1 : squelettisation par érosion (1)
Pour une forme F, on construit son squelette S_k en k-connexité érosions successives. $\overline{S_k}$ désigne le complémentaire de S_k dans l'image. On suppose que F n'a qu'une composante connexe. Le squelette est l'ensemble S_k final.

Etape A : initialistion
$S_k = F$

Etape B : érosion
$X \longleftarrow f_{12-k}\left(\overline{S_k}\right) \cap S_k$
$L \longleftarrow \emptyset$
pour chaque $p \in X$ **faire**
si $S_k - \{p\}$ **ne contient qu'une composante connexe alors**
$L \longleftarrow L \cup \{p\}$
fin si
fin pour
$S_k \longleftarrow S_k - L$

Etape C : terminaison
si $L \neq \emptyset$ **alors**
retour à l'étape B
fin si

L'ensemble X correspond au squelette de la forme en cours d'érosion. Cet algorithme est assez coûteux puisque, à chaque étape B, une passe d'image est effectuée pour construire l'ensemble $f_{12-k}\left(\overline{S_k}\right)$. L'idée est alors d'utiliser la carte de distance (définie en B.3.1) construite à l'aide du masque M_4 ou M_8 (figure B.8). Cette dernière contient pour chaque pixel de la forme F un nombre qui correspond presque à l'itération de l'étape B de l'algorithme B.4.1 dans laquelle il sera traité.

Réaliser une érosion sur une forme F revient à chercher tous les points du contour qu'on peut supprimer sans accroître le nombre de composantes connexes. Pour cela, il suffit de se limiter à un voisinage 3x3 d'un point du contour et de vérifier s'il correspond à une des configurations de la figure B.10. Ceci mène

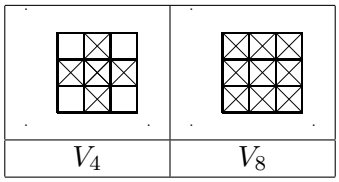


Fig. B.9: Voisinages 4-connexe et 8-connexe : le voisinage du point central correspond à l'ensemble des cases cochées.

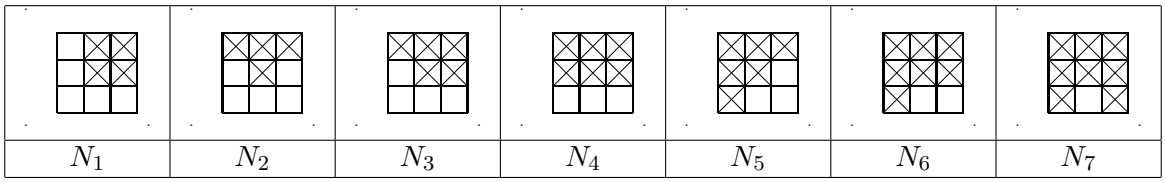


Fig. B.10: Masques de nettoyage pour construire un squelette en 4 ou 8-connexité. Pour chaque masque, l'ensemble des cases cochées correspond à des pixels appartenant à la figure F dont il faut obtenir le squelette. Si le voisinage du pixel central correspond à une de ces figures, il est nettoyé et prend la couleur du fond. Il faut également envisager les transformations (par symétrie axiale ou rotation) de ces figures, 4 images par rotation, 4 images par symétrie axiale, soit 36 figures différentes au total.

à l'algorithme suivant :

Algorithme B.4.2 : squelettisation par érosion (2)
Pour une forme F incluse dans l'image I, on construit son squelette S_4 en 4-connexité, par érosions successives. Le squelette est l'ensemble L final.

Etape A : carte de distance
On construit la carte de distance avec l'algorithme B.3.4 et le masque M_4 de la figure B.8. La forme F est constituée des pixel p pour lesquels $C^I(p) > 0$ où C est la carte de distance. Puis on construit la liste des pixels $L = (p_1, \ldots, p_n)$. Cette liste est triée par ordre de distance croissante : $i \leqslant j \Longrightarrow C^I(p_i) \leqslant C^I(p_j)$.

Etape B : érosion
$A \longleftarrow \emptyset$
pour chaque $p \in L$ **faire**
Si le voisinage du pixel p correspond à une des configurations N_1 à N_7 de la figure B.10 (ou leurs transformations par symétrie axiale ou rotation) alors :
$A \longleftarrow A \cup \{p\}$
fin pour

Etape C : suppression
si $A \neq \emptyset$ **alors**
$L \longleftarrow L - A$
L'ordre initial des pixels dans la liste L doit être conservé.
Retour à l'étape B.
fin si

Remarque B.4.3: tri de la liste L
Le tri de la liste L par ordre de distance croissante permet de supprimer d'abord les points du contour avant ceux de l'intérieur. Si ce tri n'est pas fait, le résultat n'est plus aussi parfait (figure B.11).

Remarque B.4.4: masques N_6 et N_7
Les masques N_6 et N_7 (voir figure B.10) sont nécessaires afin d'éviter que le squelette ne présente des

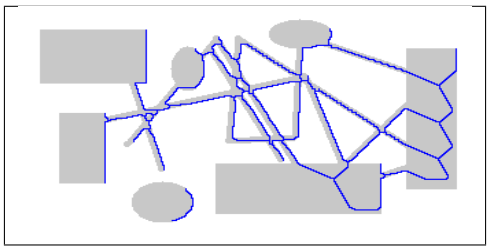


Fig. B.11: Squelette décentré obtenu lorsque l'érosion n'est pas effectuée en éliminant d'abord les pixels noirs les plus proches de pixels blancs, soit lorsque l'algorithme B.4.2 ne trie pas l'ensemble L. Ce résultat est à comparer avec celui de la figure B.1.

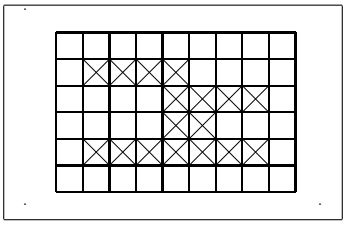


Fig. B.12: L'érosion de la forme n'est pas terminée sans l'application des masques N_6 ou N_7 de la figure B.10.

configurations comme celle de la figure B.12 où il reste une case à nettoyer.

Remarque B.4.5: quatre points résistants
L'algorithme de squelettisation B.4.2 produit une figure particulière qui peut être délicate à traiter si l'objectif est de vectoriser le squelette. La figure B.13 montre quatre branches qui se rejoignent en un bloc de quatre pixels dont on aurait préféré qu'ils ne soient qu'un.

Le passage à un squelette 8-connexité s'effectue en appliquant de nouveaux masques au squelette 4-connexe.

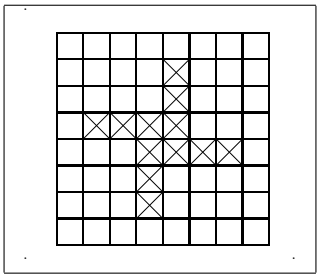


Fig. B.13: Quatre branches liées par un bloc de quatre pixels, cette figure est obtenue pour un squelette 4-connexe.

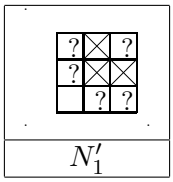


Fig. B.14: Masque de nettoyage pour construire un squelette en 8-connexité. Pour ce masque, l'ensemble des cases cochées correspond à des pixels appartenant à la figure F, les cases marquées d'un point d'interrogation peuvent être cochées ou non. Si le voisinage du pixel central correspond à cette figure (ou ses transformations par sysmétrie ou rotation), il est nettoyé et prend la couleur du fond.

Il s'agit de l'algorithme suivant :

Algorithme B.4.6 : squelettisation par érosion (3)
Pour une forme F incluse dans l'image I, on construit son squelette S_4 en 4-connexité par érosions successives. Le squelette est l'ensemble L final.

Etape A : squelette 4-connexe
On construit le squelette S_4 (4-connexe) grâce à l'algorithme B.4.2.
$L \longleftarrow S_4$

Etape B : érosion
$A \longleftarrow \emptyset$
pour chaque $p \in L$ **faire**
Si le voisinage du pixel p correspond à la configuration N_1' de la figure B.14 (ou à ses transformations) alors : $A \longleftarrow A \cup \{p\}$
fin pour

Etape C : suppression
si $A \neq \emptyset$ **alors**
$L \longleftarrow L - A$
Retour à l'étape B.
fin si

Dans les algorithmes précédents (B.4.2 et B.4.6), il faut vérifier le voisinage de chaque pixel correspond à des configurations. L'algorithme qui suit propose une expression simplifiée de ces vérifications, il est tiré

de [Marthon1979].

Algorithme B.4.7 : squelettisation de Marthon
Pour une forme F incluse dans l'image I, on construit son squelette S_k en k-connexité.
Etape A : carte de distance
On construit la carte de distance avec l'algorithme B.3.4 et le masque M_k de la figure B.8. La forme F est constituée des pixels p pour lesquels $C^I(p) > 0$ où C est la carte de distance. Puis on construit la liste des pixels $L = (p_1, \ldots, p_n)$. Cette liste est triée par ordre de distance croissante : $i \leqslant j \Longrightarrow C^I(p_i) \leqslant C^I(p_j)$.
Etape B : érosion
$A \longleftarrow \emptyset$
pour chaque $p \in L$ **faire**
Soit V_k le voisinage de la figure B.9 pour le pixel p.
$U \longleftarrow V_k \cap L$
$x \longleftarrow \sum_{p' \in U} p'_x - p_x$ et $y \longleftarrow \sum_{p' \in U} p'_y - p_y$
$z \longleftarrow |x| + |y|$
si $z = 4$ **alors**
$A \longleftarrow A \cup \{p\}$
sinon
si $z = 3$ **alors**
selon ses voisins, $A \longleftarrow A \cup \{p\}$
fin si
fin si
fin pour
Etape C : suppression
si $A \neq \emptyset$ **alors**
$L \longleftarrow L - A$
L'ordre initial des pixels dans la liste L doit être conservé.
Retour à l'étape B.
fin si

B.4.2 Erosion à partir de masques plus larges

Les érosions décrites au paragraphe B.4.1 éliminent les pixels en s'appuyant sur un voisinage $(3, 3)$ centré sur le pixel considéré, elles mènent parfois à des configurations indésirables (figure B.13) ou éliminent trop de pixels (voir figure B.15). Pour des algorithmes plus minutieux, il faut avoir recours à un voisinage plus grand. Les problèmes liés à la taille du voisinage rencontrés ici sont similaires à ceux concernant le lissage du contour évoqué au paragraphe 3.4.2 (page 51). Certains masques utilisés pour cette tâche sont définis sur des voisinages plus grands que 3x3 (voir figure 3.16, page 52) et donnent une idée de ce qu'il est possible de faire à partir des méthodes de squelettisation par érosion.

B.4.3 Ligne de crête

La carte de distance associe à chaque pixel d'une forme F la distance au pixel blanc le plus proche. Si cette distance est considérée comme une altitude, il est possible de définir les lignes de crêtes du paysage formé par la carte. Cette méthode de squelettisation est composée de deux étapes :

1. Recherche des maximas locaux : le squelette inclut les points dont l'altitude est supérieure à toutes celles de ses 4 ou 8 voisins.

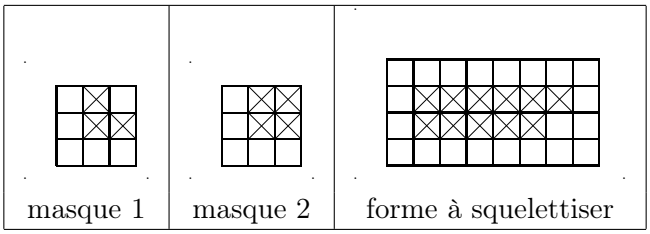


Fig. B.15: Le nettoyage de cette forme par les deux masques de gauche ne laisse que trois pixels.

2. Prolongations des lignes formées à l'étape 1 : la première étape aboutit à la formation de lignes discontinues qui doivent être prolongées afin de retrouver un squelette homotope à la forme d'origine.

Cette méthode est plus rapide qu'un algorithme basé sur des érosions succesives lorsque la forme dont il faut extraire le squelette est "épaisse" car l'algorithme se concentre tout de suite sur les points essentiels.

B.4.4 Algorithmes parallèles de squelettisation

Les algorithmes de squelettisation basés sur une érosion à l'aide de masques incluent de nombreux tests sur les pixels. En divisant cet ensemble de masques en plusieurs groupes disjoints (ou de faible intersection), il est possible de paralléliser ces algorithmes : dans ce cas, plusieurs processus - autant qu'il y a de groupes - érodent l'image, chacun capable de n'enlever que des pixels vérifiant la configuration décrite par le groupe de masques qui lui est associé. La parallélisation ne rend pas l'algorithme moins coûteux mais permet de dimininuer son temps d'exécution. L'article [ZhangY1997] s'intéresse à quatre de ces algorithmes utilisés avec deux processus. Il compare leur coût et leur redondance, qui est définie ici comme l'intersection entre les deux groupes de masques utilisés.

B.4.5 Extraction du squelette basée sur un critère de connexité

La méthode est tirée de l'article [Choi2003]. Si on considère un point P d'une forme à squelettiser, on note $Q(P)$ le point le plus proche de P appartenant au contour, on note également $(P_i)_{1\leqslant i\leqslant 8}$ les huits voisins de P. A priori, si le point P n'appartient pas au squelette, l'ensemble de points $(Q(P_i))_{1\leqslant i\leqslant 8}$ seront proches les uns des autres. En revanche, si le point P appartient au squelette, l'ensemble $(Q(P_i))_{1\leqslant i\leqslant 8}$ sera dispersé sur deux bords opposés du contour (voir figure B.16).

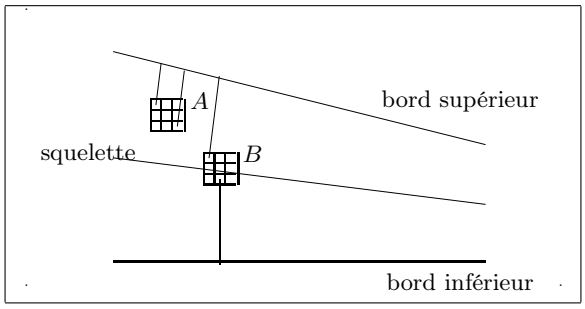


Fig. B.16: Idée sous-jacente de la squelettisation proposée par [Choi2003]. Le point A n'est pas situé sur le squelette, tous ses voisins sont plus proches du bord supérieur que du bord supérieur. A l'inverse, le point B appartient au squelette, ses voisins, selon leur position par rapport à A, sont plus proches soit du bord supérieur, soit du bord inférieur.

L'algorithme suppose de connaître pour chaque pixel de la forme le point le plus proche appartenant au contour, cette information peut être obtenue facilement à partir des cartes de distance (voir paragraphe B.3) en conservant le pixel ayant permis d'atteindre le minimum de distance (la carte de distance est ici estimée pour une forme réduite à son contour).

Algorithme B.4.8 : squelettisation (Choi2003)
La forme à squelettiser est notée F, son contour est noté $\overline{F}$ et son squelette $S(F)$. On note $\rho > 0$ un paramètre réel et positif. On note également $X(P)$ et $Y(P)$ respectivement l'abscisse et l'ordonnée de P.

Etape A : carte de distance
Calcul d'une carte de distance permettant d'associer à chaque pixel $P \in F$ le point $Q(P) \in \overline{F}$.

Etape B : squelettisation
Pour tout point $P \in F$, soit $(P_i)_{1 \leqslant i \leqslant 8}$ les voinsins de P en 8-connexité, on définit :

$$\forall i \in \{1, \ldots, 8\}\,,\ Q_i = Q(P_i) + P - P_i$$

Alors P appartient au squelette s'il existe $i \in \{1, \ldots, 8\}$ tel que :

$$\|Q_i - Q(P)\|^2 \geqslant \rho \tag{B.2}$$

$$\text{et } \|Q_i\|^2 - \|Q\|^2 \leqslant \max\{X(Q_i - Q)\,,\ Y(Q_i - Q)\} \tag{B.3}$$

Le coût de cet algorithme est linéaire par rapport au nombre de pixels de la forme à squelettiser, ce qui représente un avantage certain par rapport aux algorithmes basés sur une érosion (voir paragraphe B.4.1). Il reste à ajuster la valeur ρ, petite pour des squelettes fournis, grande pour des squelettes dégarnis (voir figure B.17). Sans la seconde condition de l'algorithme, le squelette obtenu a trois pixels d'épaisseur, cette condition permet de ramener cette épaisseur sur un pixel dans la majorité des cas. Il est parfois souhaitable d'affiner le résultat par une étape d'érosion afin d'obtenir l'epaisseur ou la connexité voulues.

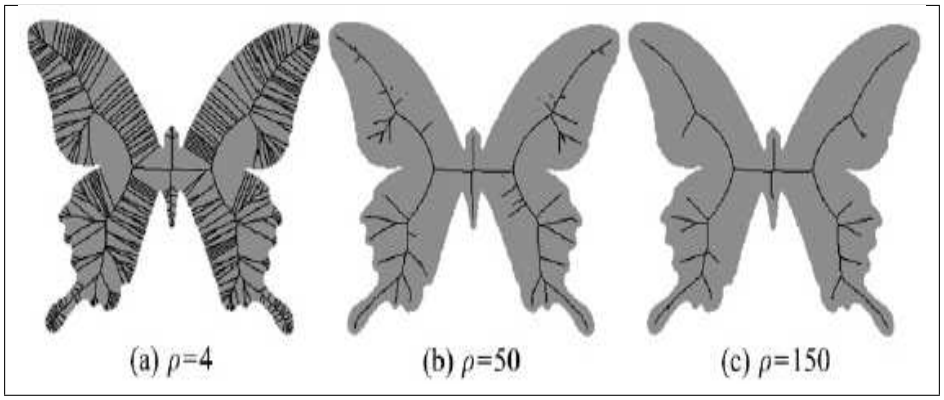


Fig. B.17: Squelettisation à partir d'un critère de connexité, squelettes obtenus pour différentes valeurs du paramètre ρ (figure extraite de [Choi2003]).

Remarque B.4.9: pertes de connexité
Cet algortihme n'est pas bien adapté à la squelettisation des caractères manuscrits. Ces formes sont souvent très fines et le résultat final présente de nombreuses barbules et quelques pertes de connexité dues à la condition (B.2) qui détruit le squelette dans les zones dont l'épaisseur est trop fine.

B.4.6 Squelettisation à partir de filtre de Gabor

L'article [Su2003] propose une méthode fondée sur les filtres de Gabor et l'applique au cas des caractères chinois. Ces caractères sont principalement composés des traits verticaux, horizontaux et diagonaux, une première étape de filtrage permet d'extraire ces quatre types de traits comme le montre la figure B.18, chaque trait est ensuite vectorisé ce qui permet d'obtenir une première approximation du squelette (avant-dernière colonne de la figure B.18). Les traits sont ensuite reconnectés entre eux pour obtenir le squelette final selon des critères de proximités et de direction. L'avantage de cette méthode est sa grande stabilité par rapport au bruit comme le montre les deux dernières lignes de la figure B.18. Les post-traitements regroupant le nettoyage et la connexion des traits succèdent au filtrage de Gabor. Comme ils sont indépendants de la squelettisation, seul le filtrage de Gabor sera décrit.

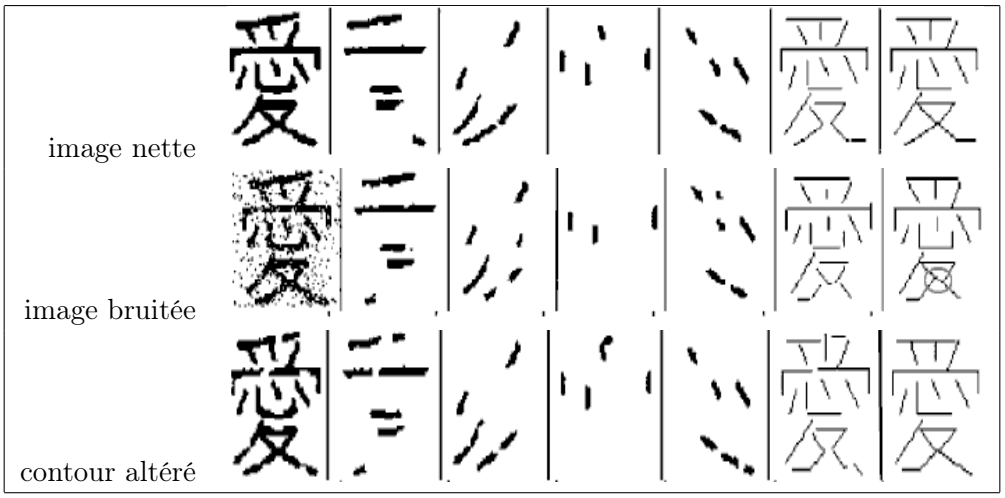


Fig. B.18: Extraction des traits d'une image selon quatre directions, verticale, horizontale, diagonales, à l'aide de filtres de Gabor, figure extraite de [Su2003]). La première ligne montre le résultat sur une image nette, la seconde ligne sur une image bruitée aléatoirement, la dernière ligne, sur une image dont le contour a été altéré.

On note $i(x,y) \in [0,1]$ l'intensité de l'image, et $h(x,y)$ le filtre de Gabor défini par :

$$h(x,y) = \exp\left(-\pi \frac{x^2+y^2}{\sigma^2}\right) \exp\left(2i\pi f \ (x\cos\theta + j\sin\theta)\right) \tag{B.4}$$

La réponse de l'image au filtre de Gabor est notée $I(x,y)$:

$$I(x,y) = |i(x,y) \otimes h(x,y)| \tag{B.5}$$

Les paramètres utilisées sont les suivants :

$$\sigma = \frac{\sqrt{2}}{f} \quad f = 0,9857\frac{e}{dh}$$

e est l'épaisseur moyenne des traits (voir paragraphe 3.4.4, page 53), h est la hauteur de l'image, d est la densité de l'image où le rapport entre le nombre de pixels noirs et la taille de d'image. L'angle θ prend quatre valeurs pour les quatre directions désirées : $\theta \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right\}$, le résultat de ces quatre filtres est illustré par la figure B.19.

Fig. B.19: Réponses des quatres filtres de Gabor correspondant aux quatre directions, 0°, 45°, 90°, 135°, figure extraite de [Su2003].

Ces quatre images sont ensuite binarisées pour obtenir les quatre images $K_n(x,y)_{1\leqslant n\leqslant 4}$:

$$\forall (x,y),\ K_n(x,y) = \mathbb{1}_{\{I_n(x,y)\geqslant\alpha\}} \tag{B.6}$$

Le seuil α est calculé de manière itérative de façon à ce que les quatre images K_n ne contiennent pas d'informations redondantes, deux autres images sont alors construites :

$$\forall (x,y),\ M_\alpha(x,y) = \sum_{n=1}^{4} K_n(x,y) \tag{B.7}$$

$$\forall (x,y),\ N_\alpha(x,y) = \sum_{n=1}^{4} \mathbb{1}_{\{M_\alpha(x,y)\geqslant 1\}} \tag{B.8}$$

Les quatre images K_n permettent de reconstruire l'image originale, deux types d'erreurs sont alors quantifiés, l'erreur en perte $E_p(\alpha)$ et l'erreur de recouvrement $E_r(\alpha)$:

$$E_p(\alpha) = \frac{\sum\limits_{x,y} |N_\alpha(x,y) - i(x,y)|}{\sum\limits_{x,y} i(x,y)} \qquad E_r(\alpha) = \frac{\sum\limits_{M_\alpha(x,y)>1} M_\alpha(x,y) - i(x,y)}{\sum\limits_{x,y} i(x,y)} \tag{B.9}$$

L'erreur globale est définie comme la moyenne des deux : $E(\alpha) = \frac{1}{2}(E_r(\alpha) + E_p(\alpha))$. Le seuil α est choisi comme la limite de la suite α_t définie par :

$$\alpha_0 \in [0,1] \tag{B.10}$$

$$\alpha_{t+1} = \alpha_t + \left[\mathbb{1}_{\{E(\alpha_t)\geqslant E(\alpha_{t-1})\}} - \mathbb{1}_{\{E(\alpha_t)<E(\alpha_{t-1})\}}\right] \tanh\left(\frac{E(\alpha)}{2}\right) \tag{B.11}$$

Les images obtenus après ce seuillage ne sont pas encore parfaites (voir figure B.20), elles sont ensuite nettoyées des trop petits segments en tenant compte des attributs tels que la surface et la longueur, leur présence sur plus d'une image.

B.5 Squelettisation d'une forme vectorielle

Ces méthodes diffèrent des précédentes car elles utilisent uniquement le contour de la forme à squelettiser, ce dernier étant décrit comme une succession de segments. Les méthodes d'érosion, quant à elles, partent

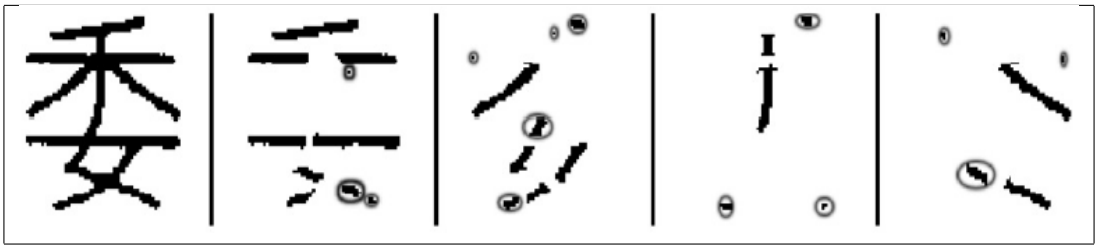


Fig. B.20: Réponses seuillées des quatres filtres de Gabor correspondant aux quatres directions verticale, horizontale, diagonales, figure extraite de [Su2003]. Les petits segments entourés vont être supprimés par le nettoyage.

d'une forme décrite par un ensemble de pixels connexes. Il est bien sûr possible d'appliquer les méthodes vectorielles à ces ensembles de pixels en les réduisant à leur contour. Les pixels du contour obtenus sont alors les sommets des segments. Afin de réduire la complexité des méthodes vectorielles, l'ensemble des pixels formant le contour est souvent réduit. Les sommets utilisés pour la squelettisation sont répartis à égale distance le long de la courbe (voir figure B.21) ou celle-ci peut-être vectorisée de manière à regrouper ensemble des segments successifs colinéaires (voir figure B.22).

B.5.1 Diagramme de Voronoï

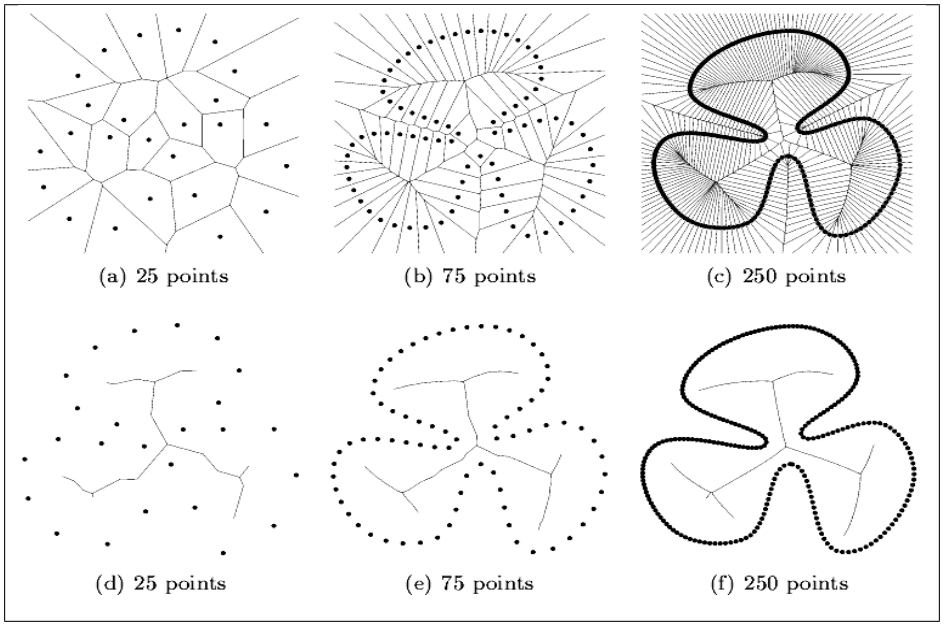


Fig. B.21: Squelettisation à partir d'un graphe de Voronoï, figure extraite de [Attali1995].

La définition de l'axe médian (B.1.2) fait intervenir le centre de boules tangentes en au moins deux points du contour de cette forme. Par conséquent, en considérant deux points du contour, le squelette est susceptible de passer par la médiatrice de ces deux points. Le diagramme de Voronoï est justement un ensemble de médiatrices. Il suffit alors de disposer des points éparpillés sur le contour, de déterminer le diagramme de Voronoï puis de l'élaguer pour ne garder que les segments de médiatrice empruntés par le squelette. Cette idée a été developpée dans [Ogniewicz1992], [Ogniewicz1995], ou encore [Attali1995]. La figure B.21 montre que le squelette est plus précis lorsque les points sur le contour sont plus nombreux mais évidemment plus lent à calculer. Une fois que le diagramme de Voronoï associé à une forme est construit,

le squelette est tout simplement constitué des seuls segments inclus dans l'intérieur de cette forme.

B.5.2 Réseau bissecteur

La squelettisation par réseau bissecteur est assez proche de celle developpée à partir d'un diagramme de Voronoï (voir [Cloppet2000]). Une forme est définie par son contour lui-même défini comme une succession d'arêtes. Les premiers nœuds du réseau bissecteur sont formés par les bissectrices de chaque angle. Lorsque deux bissectrices s'interceptent, elles forment un angle dont on peut à nouveau tracer la bissectrice (voir figure B.22). Ce graphe est similaire à celui obtenu par le diagramme de Voronoï, le squelette est simplement une sous-partie de ce diagramme.

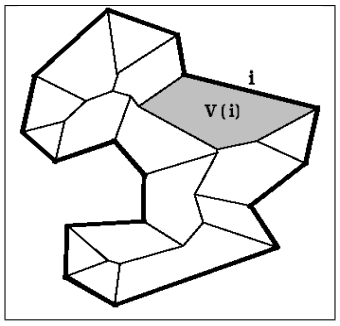


Fig. B.22: Squelettisation à partir d'un réseau bissecteur, figure extraite de [Cloppet2000].

B.6 Affinement du résultat

B.6.1 Nettoyage des barbules

Le squelette d'une image est sa représentation en "fil de fer", celle-ci peut-être plus ou moins précise. Après la squelettisation, un grand nombre de petits segments du squelette peuvent s'avérer non pertinents comme le montre la figure B.23. Le squelette d'un mot manuscrit devrait être proche du mouvement du stylo. Toutefois, la squelettisation conserve un ensemble de petits segments sans importance dont la suppression ne modifie pas le nombre de composantes connexes. Nettoyé de ces barbules, le squelette est en quelque sorte plus "lisible" mais il n'y a pas de règles pour ce nettoyage, il dépend à la fois de l'algorithme de squelettisation employé et de la précision désirée.

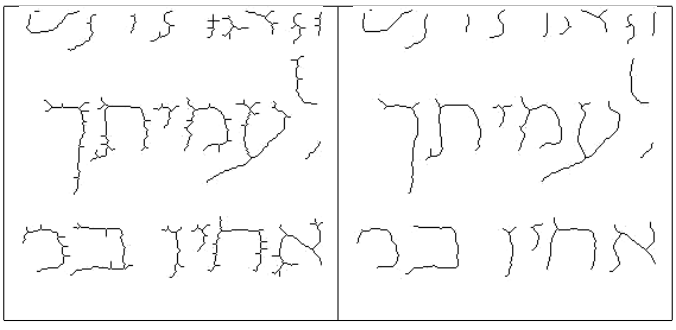


Fig. B.23: La seconde image représente le squelette de la première image nettoyé de ses barbules, la seconde image contient simplement l'information pertinente.

Ce nettoyage peut s'appuyer sur des critères géométriques tel que celui proposé dans [Jang1992] qui mesure

le rapport entre la forme F à squelettiser et l'aire formée par l'ensemble G des boules maximales incluses dans F dont le centre appartient au squelette. Par définition, $G \subset F$, le critère de [Jang1992] est égal à :

$$c_J = \frac{aire\,(G)}{aire\,(F)} \leqslant 1 \tag{B.12}$$

Le squelette est rogné à ses extrémités tant que le critère c_J est supérieur à un certain seuil. Un autre critère provient de [Huang2003] qui compare les longueurs (en pixel) du squelette et du contour :

$$c_H = \frac{aire\,(squelette)}{aire\,(contour)} \leqslant 1 \tag{B.13}$$

B.6.2 Squelette d'une boucle

Cet article [Huang2003] propose également d'effectuer l'érosion de la forme tant que le critère c_H (B.13) est supérieur à un certain seuil. Par conséquent, pour un seuil bien ajusté, les zones peu épaisses sont bien squelettisées tandis que les zones épaisses sont un compromis entre squelette et contour, la figure B.24 montre le résultat qu'il est possible d'obtenir avec ce genre de méthode. Le squelette du chiffre "6" est obtenu avec sa boucle bien que celle-ci soit comblée de pixels noirs.

Fig. B.24: Le squelette de cet image est une étape intermédiaire entre le contour et le véritable squelette. Ce résultat est pourtant proche de celui qu'il faut obtenir puisque la boucle du chiffre "6" disparaît avec un algorithme de squelettisation classique.

B.6.3 Améliorer la représentation des intersections

Lors de l'extraction du squelette d'un caractère, les intersections entre deux segments de droites sont souvent scindées comme le montre la figure B.25b. La méthode développée dans [Zhong1999] propose de corriger le squelette obtenu en figure B.25b pour aboutir à celui B.25c. La méthode est appliquée sur des caractères chinois qui présentent souvent des intersections croisant un trait horizontal et un trait vertical. En explorant l'image initiale selon des lignes parallèles aux diagonales, il est possible de marquer quatre types de points caractéristiques d'une intersection (voir figure B.25d qui indique le début ou la fin d'un embranchement).

Le squelette est alors corrigé en effaçant tout d'abord la partie incluse entre ces quatre points puis en faisant converger vers un même et unique point les quatre extremités du squelette le reliant à cette zone.

Les points caractéristiques sont détectés à l'aide des "run-length" définis dans l'article comme des segments de points contigüs ou ensembles de pixels noirs contigüs positionnés sur la même ligne. Les embranchements sont détectés en étudiant le nombre de "run-length" et leur chevauchement. Un "run-length" chevauchant deux "run-length" de la ligne suivante désignent une divergence. La configuration opposée désigne une convergence.

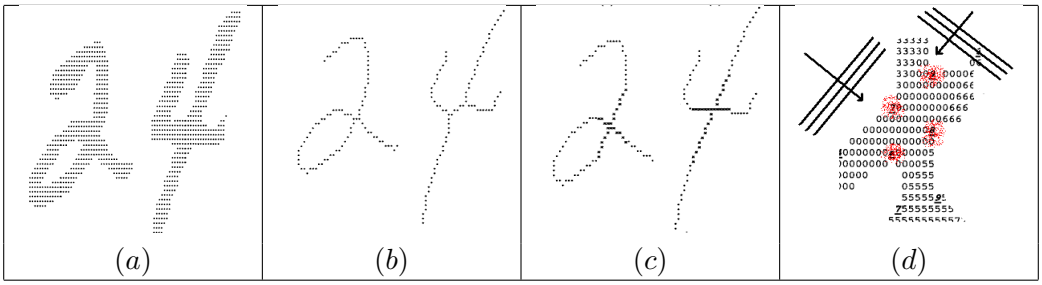


Fig. B.25: Correction du squelette afin de mieux représenter les intersections, figures extraites de [Zhong1999]). L'image (d) montre les quatre points cardinaux d'une intersection. L'objectif de la méthode est de passer l'image (b) à l'image (c). Avant l'érosion, des points cardinaux sont détectés, symbolisant chacun un embranchement. Quatre d'entre eux à la figure (d) permettent de repérer une intersection, le squelette à l'intérieur de la zone encadrée par ces quatre points ne sera pas issu d'une érosion mais deux droites joignant les quatre extrémités du squelette situées aux limites de cette zone.

B.6.4 Modéliser les intersections dans les caractères

L'article [L'Homer2000] propose une modélisation intéressante du squelette. Cet article s'intéresse tout particulièrement à la squelettisation de caractères manuscrits et cherche à extraire un squelette décrit comme une fonction dépendant du temps et proche de l'évolution du stylo sur la feuille de papier. L'auteur construit un modèle permettant de décomposer le tracé d'une lettre sous forme d'arcs continus et réguliers (la dérivée est bornée) représentés dans la troisième ligne de la figure B.26.

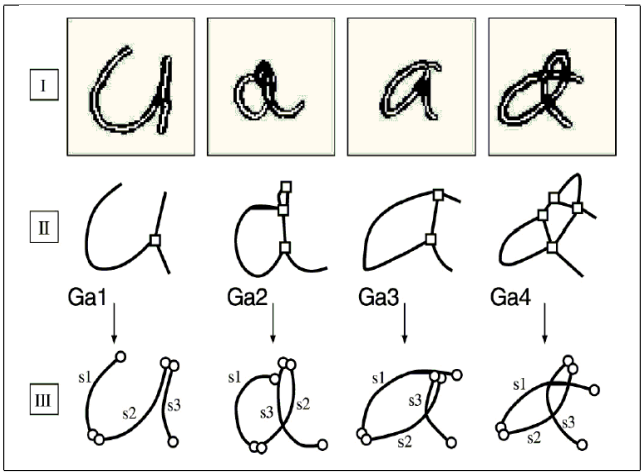


Fig. B.26: Figure extraite de [L'Homer2000] représentant la modélisation de différentes lettres "a". Les arcs formant ces lettres peuvent se rejoindre dans un point de rebroussement (premier colonne), s'intercepter comme pour la barre d'un "T" (seconde colonne), se croiser (dernière colonne).

L'algorithme détecte les arcs réguliers (première ligne de la figure B.26) ainsi que leurs extremités devant être jointes. L'attrait de cet article réside essentiellement dans la façon de réaliser ces jointures. Certains croisements représentent deux traits qui se rejoignent à un point de rebroussement (première lettre "a" de la figure B.26), d'autres représentent des traits qui se croisent (dernière lettre "a" de la figure B.26). La liste des modèles de jointures est illustrée par la figure B.27. Le modèle le plus probable tend à conserver les courbes les plus régulières possibles.

Cette représentation du squelette est conçue pour des caractères manuscrits et propose une description du squelette plus évoluée que les précédentes. La détection de figures particulières incluses dans les caractères se contente souvent des boucles. Cet article propose une description sous forme d'arcs réguliers ainsi

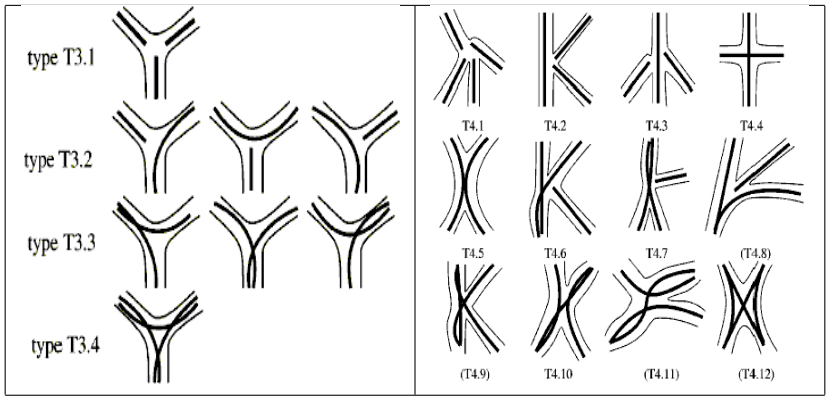


Fig. B.27: Figure extraite de [L'Homer2000] représentant les différentes types des jointures entre arcs. Un même trait peut cacher deux passages du stylo et c'est ce que ces modèles tentent de détecter. La première image présente les huit possibilités de branchements lorsque trois arcs se croisent. La seconde image présente les seize possibilités de branchements lorsque quatre arcs se croisent.

que les types d'intersections les reliant entre eux. Il pourrait être intéressant d'étudier l'apport de telles caractéristiques pour la reconnaissance des caractères.

Cette méthode propose à la fois une représentation paramétrée du squelette ainsi qu'une amélioration de la description des intersections. Cette représentation sous forme d'arcs est guidée par la forme des caractères et même si la plupart des méthodes abordées dans ce document ont traits à ce domaine, il est possible d'adopter des représentations paramétriques du squelette plus simple mais plus générales, par exemple, sous forme de droites. Ce processus s'appelle la vectorisation et est présentée dans les paragraphes qui suivent.

B.7 Post-traitements

B.7.1 Vectorisation du squelette

Le squelette obtenu est une suite de pixels. Il peut être intéressant de le vectoriser, autrement dit de le décrire à l'aide de segments de droites. Cette vectorisation peut consister en la recherche des droites qui auraient permis le tracé du squelette comme le suggère les thèses [Reveillès1991] et [Vittone1999]. Ces

travaux utilisent la définition suivante (extraite de [Reveillès1991]) :

Définition B.7.1 : droite discrète
Soient $(a, b, r) \in \mathbb{Z}^2$ et $\omega \in \mathbb{N}^*$. Une droite de vecteur directeur (b, a) avec $(a, b) \neq (0, 0)$ et $pgcd\,(a, b) = 1$, de paramètre de translation r et d'épaisseur arithmétique ω est l'ensemble noté $D\,(a, b, r, \omega)$ des points $(x, y) \in \mathbb{Z}^2$ satisfaisant l'inégalité :

$$0 \leqslant ax + by + r < \omega$$

De plus :

si $1 \leqslant w \leqslant max\,(\lvert a\rvert, \lvert b\rvert)$	alors	la droite n'est pas connexe et est dite déconnectée.
si $\omega = max\,(\lvert a\rvert, \lvert b\rvert)$	alors	la droite est 8-connexe.
si $max\,(\lvert a\rvert, \lvert b\rvert) \leqslant \omega\,\lvert a\rvert + \lvert b\rvert$	alors	la droite est 8-connexe et 4-connexe par moment.
si $\omega = \lvert a\rvert + \lvert b\rvert$	alors	la droite est 4-connexe.
si $\omega > \lvert a\rvert + \lvert b\rvert$	alors	la droite est épaisse

D'un point de vue plus pragmatique, l'article [Freeman1970] décrit les trois propriétés vérifiées par le code de Freeman d'une ligne 8-connexe :

1. Au plus deux directions peuvent être présentes dans le code et ne peuvent différer que d'une unité modulo 8.
2. Une des deux directions apparaît toujours de manière isolée.
3. La direction isolée apparaît de manière uniforme dans le code.

La première propriété permet d'isoler les portions de code susceptibles de représenter des droites, les deux suivantes permettent d'estimer les paramètres de son équation. En effet, les occurences des deux directions intervenant dans la description mènent directement au vecteur directeur de la droite. Les algorithmes développés dans [Vittone1999] ou [Breton2002] utilisent la définition B.7.1 et retrouvent les segments de droite au pixel près comme le montre la figure B.28 : la droite est recouverte par l'ensemble de pixels vectorisés par cette droite.

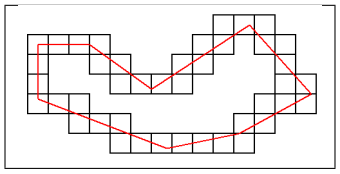


Fig. B.28: Vectorisation d'un contour : figure extraite de [Breton2002], le contour vectorisé est recouvert par l'ensemble de pixels formant le contour.

Une autre méthode permet de vectoriser un arc quelconque de manière approchée. Si cet arc est proche d'une droite alors la corde reliant ses deux extrémités est une bonne approximation (voir figure B.29). En revanche, si cet arc n'est pas une droite, la corde est une mauvaise approximation, l'arc est alors divisé en deux parties dont l'extrémité commune est le point de l'arc le plus éloigné de sa corde. Ceci mène à

l'algorithme suivant :

Algorithme B.7.2 : vectorisation approchée
Soit un arc k-connexe défini par une suite de pixels $(p_1, \ldots, p_n)$ vérifiant :

$$\forall i \in \{2, \cdots, n\}, \ p_{i-1} \in V_k(p_i)$$

On calcule le critère c défini par :

$$c = \max_{i \in \{1, \cdots, n\}} d(p_i, D(p_1, p_n))$$

où $d(p_i, D(p_1, p_n))$ est la distance du point p_i à la droite passant par les points p_1 et p_n. Si c est trop grand (supérieur à un seuil), alors l'arc est divisé en deux parties $(p_1, \ldots, p_j)$ et $(p_j, \ldots, p_n)$ où p_j vérifie :

$$p_j \in \arg\max_{i \in \{1, \cdots, n\}} d(p_i, D(p_1, p_n))$$

L'arc est ainsi découpé jusqu'à ce que plus aucune division ne soit possible.

Remarque B.7.3: choix de p_j
Dans le précédent algorithme, p_j est un des points les plus éloignés de la corde. Si plusieurs points sont à égale distance de la corde, le point p_j choisi est de préférence celui qui est le plus au centre de l'arc.

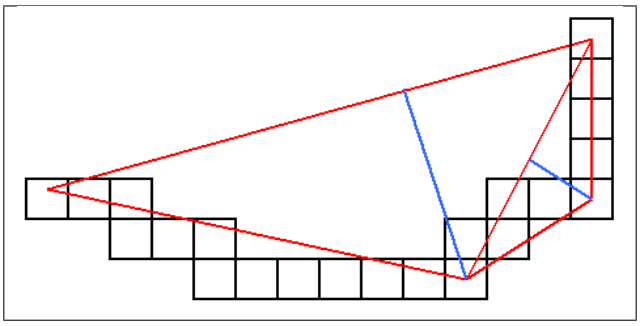


Fig. B.29: Vectorisation approchée d'un arc : l'arc est divisé en deux parties dont l'extrémité commune est le point de l'arc le plus éloigné de sa corde.

L'article [Yeung1996] effectue ce travail dans le cadre de caractères chinois avec une méthode différente, les arcs du squelette sont scindés en plusieurs segments de droites en détectant les changements abruptes d'orientation, le découpage n'est pas plus conditionné par des contraintes de distances mais des contraintes angulaires. Comme ce travail s'articule autour de la reconnaissance de caractères chinois comprenant souvent plusieurs composantes connexes, il propose également une méthode pour connecter les squelettes de deux composantes connexes lorsque certains configurations apparaissent - par exemple la lettre "T" dont la barre est dissociée de son support -.

Il est possible d'aller plus loin dans la vectorisation du squelette et c'est ce que propose l'article [Chakravarthy2003]. Des points caractéristiques sont détectés et étiquetés le long du squelette. La figure B.30 illustre cette description sous forme de graphe à l'aide des cinq points caractéristiques parmi douze possibles :

- A : (Angle), deux arcs se rejoignent pour former un point de rebroussement.
- B : (Bump), milieu d'un arc courbe.
- C : (Cusp), jonction d'une courbe et d'un segment de droite.

– T : (T point), un segment de droite vient en intercepter un autre en son milieu.
– P : (Peck), un point de rebroussement situés au milieu d'un segment de droite.

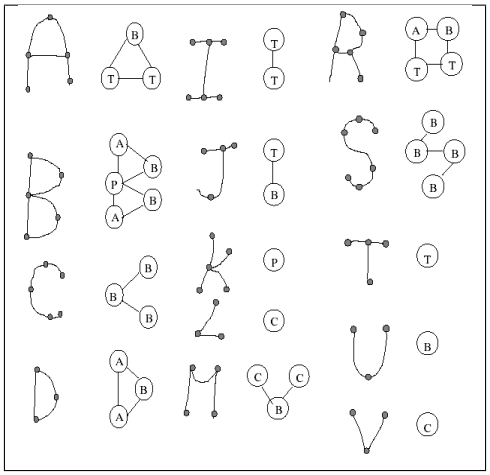


Fig. B.30: Vectorisation étendue du squelette : le squelette est vectorisé et chaque arc reçoit une étiquette choisie (A,B,C,T,P).

Ces descriptions complexes d'un squelette sous forme de graphe tendent à tirer le plus d'informations possibles de l'image elle-même de manière à pouvoir, depuis cette structure, identifier la forme représentée. Cette identification peut être effectuée grâce à une distance d'édition entre graphe, celui obtenu et un modèle représentant au mieux la forme à identifier.

B.7.2 Vectorisation et intersection

L'article [Abuhaiba1996] suppose que lorsque deux traits s'interceptent comme c'est souvent le cas pour une image contenant de l'écriture, la zone de l'intersection est plus épaisse que les zones où seul un trait apparaît. La figure B.31 illustre ceci dans le cas d'une étoile. Le squelette sans aucun post-traitement contient six points reliant trois branches. A partir d'une estimation de l'épaisseur du trait (voir paragraphe 3.4.4, page 53), il est possible de déterminer la zone d'attraction de l'intersection, zone où la carte des distances contient des valeurs supérieures à cette épaisseur.

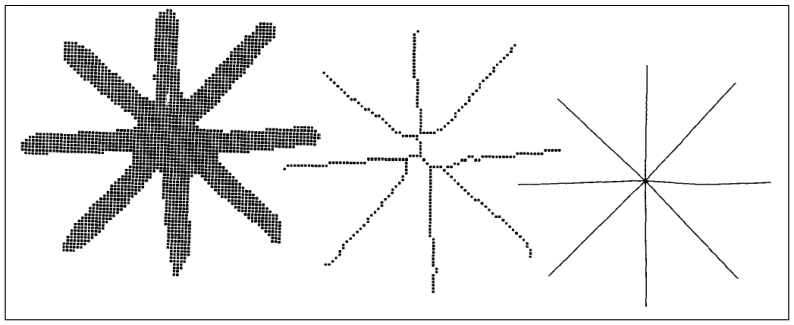


Fig. B.31: Figure extraite de [Abuhaiba1996], l'épaisseur du trait est plus épais aux intersections, la vectorisation utilise ces zones étendues pour relier entre eux les arcs du squelette. La première image est l'image originale, la seconde représente le squelette, la troisième le squelette vectorisé.

Les six points reliant chacun trois branches appartiennent tous à cette zone d'attraction et sont alors considérés comme étant une seule et même intersection. La vectorisation de ce squelette associe donc les huit segments sortant de la zone d'attraction à un seul point de recoupement.

B.7.3 Squelette d'une image de texte

Dans certains cas, certains a priori permettent d'améliorer la qualité du résultat. C'est le cas de la méthode développée au paragraphe N.4.2, page 441 qui tient compte du fait que l'image est celle d'un ou plusieurs mots. L'hypothèse simplificatrice est dans ce cas une relative homogénéité de l'épaisseur de la forme à squelettiser.

B.7.4 Appariement squelette - image originale

La segmentation en graphèmes utilise le squelette afin de découper l'image d'un mot en lettres ou morceaux de lettres. Il s'agit maintenant de propager ce découpage à l'ensemble de l'image, donc de retrouver de quelle partie de la forme initiale un morceau est le squelette. Il est possible d'utiliser la carte de distance définie en B.3.1. Pour chaque pixel noir, le pixel du squelette qui en est le plus proche apparaît en se déplaçant dans la carte de distance selon la plus grande pente. Il suffit de réitérer ce procédé pour chaque pixel à apparier.

Une autre approche permet de résoudre un problème plus général. On suppose que l'image contient un ensemble de pixels $(p_1, \ldots, p_n)$ répartis en C classes. Chaque pixel p_i est donc étiqueté par $c_i \in \{1, \cdots, C\}$. Pour un pixel p quelconque de l'image, le point le plus proche dans la suite $(p_1, \ldots, p_n)$ détermine sa classe. L'algorithme qui suit permet d'effectuer cet étiquetage de manière analogue à l'algorithme B.3.4 utilisé pour calculer une carte de distance.

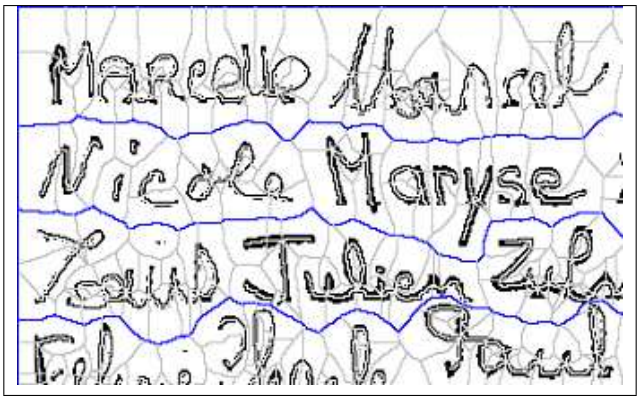


Fig. B.32: Appariement : les frontières (gris pâle et foncé) délimitent les zones de pixels (noirs) appariés au même morceau du squelette.

Algorithme B.7.4 : appariement
Soit $P = (p_1, \ldots, p_n)$ une suite de points, et $(c_1, \ldots, c_n) \in \{1, \cdots, C\}^n$ leurs classes associées. On note $D(x,y)$ la distance au point le plus proche de l'ensemble P et $C(x,y)$ la classe de ce point. On impose que $D(x,y) = \infty$ si $(x,y) \notin \{1, \cdots, X\} \times \{1, \cdots, Y\}$.

Etape A : première passe d'image
pour $y = 1$ **à** Y **faire**
pour $x = 1$ **à** X **faire**

$$D(x,y) \longleftarrow \begin{cases} 0 \text{ si } \exists i \text{ tel que } p_i = (x,y) \\ \min\left\{D[(x,y) - v] + M_{v_x,v_y} \mid v \in V_h(k)\right\} \text{ sinon} \end{cases}$$

$$C(x,y) \longleftarrow \begin{cases} \text{n'est pas défini si } D(x,y) = \infty \\ c_i \text{ si } \exists i \text{ tel que } p_i = (x,y) \\ C((x,y) - v^*) \\ \quad \text{où } v^* \in \arg\min\left\{D[(x,y) - v] + M_{v_x,v_y} \mid v \in V_h(k)\right\} \text{ sinon} \end{cases}$$

fin pour
fin pour

Etape B : seconde passe d'image
pour $y = Y$ **à** 1 **faire**
pour $x = X$ **à** 1 **faire**

$$D(x,y) \longleftarrow \min\left\{D(x,y), \min\left\{D[(x,y) - v] + M_{v_x,v_y} \mid v \in V_b(k)\right\}\right\}$$

$$C(x,y) \longleftarrow C((x,y) - v^*) \quad \text{où } v^* \in \arg\min\left\{D[(x,y) - v] + M_{v_x,v_y} \mid v \in V_b(k) \cup \{(0,0)\}\right\}$$

fin pour
fin pour

Remarque B.7.5: lien avec le diagramme de Voronoï
Si pour chaque point p_i, $c_i = i$, alors cet algorithme aboutit à la construction de régions de "Voronoï" qui serviront à construire le diagramme illustré dans la figure B.21. Le squelette sera constitué des segments séparant deux pixels appartenant à des régions différentes.

B.7.5 Construction d'un graphe pour une classification

L'article [Ruberto2004] propose la construction d'un graphe résumant le squelette. Les arcs représentent des parties du squelette tandis que les nœuds sont ses points singuliers (voir figure B.33).

La méthode développée dans cet article utilise un tel graphe qui est d'abord nettoyé des petits arcs ou barbules [1]. Il propose ensuite d'associer à chaque arc des caractéristiques qui sont utilisées afin d'effectuer une tâche de reconnaissance via le calcul d'une distance entre graphe, celle-ci permettant de comparer le squelette de deux formes entre elles.

Le graphe obtenu contient une liste d'arcs $(A_1, \ldots, A_n)$, la longueur l du squelette est la somme des longueurs de chaque arcs : $l = \sum_{i=1}^{n} l(A_i)$. Les arcs A_i dont la longueur vérifie $l(A_i) < \alpha l$ et ne contenant aucune extrémité sont supprimés puis ajoutés aux arcs auxquels ils sont connectés. α est choisi égal à 5%.

Chaque arc S est ensuite décrit par six caractéristiques. S est défini par ses deux extrémités (x_1, y_1), (x_2, y_2) et une fonction $t \in [0,1] \longrightarrow (x(t), y(t))$. Ces six caractéristiques sont données par la table B.2.

L'auteur de l'article [Ruberto2004] propose d'utiliser ce graphe "attribué" afin de calculer une distance entre deux formes pour lesquels ce graphe G aura été préalablement estimé. Ce graphe inclut un ensemble

1. Annexes : voir paragraphe B.6.1, page 175

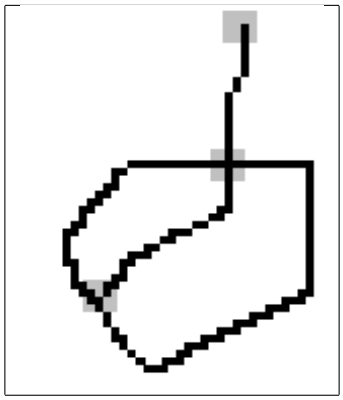


Fig. B.33: Ce squelette présente trois points singuliers : le premier est une extremité, le second est la jonction de quatre arcs, le dernier est la jonction de trois arcs.

d'arêtes décrites par les caractéristiques de la table B.2 et n nœuds qui sont les points singuliers du squelette. On définit $A = (a_{ij})_{1 \leqslant i,j \leqslant n} \in [0,1]^{n^2}$ la matrice d'adjacence du graphe G, le graphe est donc entièrement défini par $G = \{A, (v_{i,j,1}, \ldots, v_{i,j,6}) \mid 1 \leqslant i, j \leqslant n\}$. La distance entre deux graphes G_1 et G_2 est définie par :

$$d(G_1, G_2, \alpha) = \inf \left\{ E\left(G_1, G, M = (m_{ik})_{\substack{1 \leqslant i \leqslant n_1 \\ 1 \leqslant k \leqslant n_2}}, \alpha\right) \;\middle|\; \begin{array}{l} \forall i, k,\ m_{ik} \in \{0,1\} \\ \forall k,\ \sum_{i=1}^{n_1} m_{ik} \leqslant 1 \\ \forall i,\ \sum_{k=1}^{n_2} m_{ik} \leqslant 1 \end{array} \right\} \tag{B.14}$$

$$\text{avec } E(G_1, G_2, M, \alpha) = -\frac{1}{2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} \sum_{j=1}^{n_2} \sum_{l=1}^{n_2} m_{ik}\, m_{jl}\, e(i \to j,\ k \to l) + \alpha \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} m_{ik}\, e(i,k)$$

α est un terme permettant d'ajuster la prépondérance de l'association entre les nœuds par rapport à celle entre les arêtes. La fonction $e(i \to j,\ k \to l)$ mesure la vraisemblance de l'association entre l'arête $i \to j$ du premier graphe et l'arête $k \to l$ du second graphe tandis que $e(i,k)$ mesure la vraisemblance de l'association entre le nœud i de premier graphe et le nœud j du second graphe. La première fonction est définie comme suit :

$$e(i \to j,\ k \to l) = \begin{cases} 0 & \text{si } a^1_{ij} a^2_{kl} = 0 \\ \sum_{d=1}^{6} L_d \left(1 - \left| \frac{v_{i,j,d,1}}{L_d} - \frac{v_{k,l,d,2}}{L_d} \right| \right) & \text{sinon} \end{cases} \tag{B.15}$$

$$\text{avec } L_d = \frac{1}{2} \max \left\{ \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} v_{i,j,d,1},\ \sum_{k=1}^{n_2} \sum_{l=1}^{n_2} v_{k,l,d,2} \right\}$$

La fonction $e(i,k)$ est construite de manière analogue en prenant comme caractéristique l'épaisseur au point singulier par exemple. Le problème de minimisation est malheureusement NP-complet mais il peut être résolu selon une méthode approchée appelée "affection graduée" développée dans [Gold1996]. En définitive, la distance d définie en (B.14) permet d'effectuer une classification par plus proches voisins. Etant donné son coût élevé, il est préférable d'utiliser d'éviter un trop grand nombre de calculs par le biais de méthodes comme celles développées en annexe H.

variation de la courbure moyenne, soit les valeurs extrêmes de la fonction c	$v_1 = \max_{t\in[0,1]} c(t) - \min_{t\in[0,1]} c(t)$ où $c(t) = \frac{x'y''-x''y'}{((x')^2+(y')^2)^{\frac{3}{2}}}$
l'orientation de l'arc par rapport à celle du squelette	$v_2 = \arctan \frac{y_2-y_1}{x_2-x_1}$
la taille de l'arc par rapport à celle du squelette	$v_3 = \frac{l(S)}{l}$ où $l(S)$ est la longueur de l'arc et l longueur du squelette
la "raideur" de l'arc	$v_4 = \frac{l(S)}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}$
la variation de l'épaisseur distance le long de l'arc	$v_5 = \max_{t\in[0,1]} e(t) - \min_{t\in[0,1]} e(t)$ où $e(t)$ est l'épaisseur de la forme le long de l'arc, estimée par exemple à l'aide d'une carte de distance (voir paragraphe B.3)
la taille de la région R par rapport à celle du squelette	$v_6 = \frac{A(S)}{A}$ où $A(S)$ est la surface de la partie de la forme dont l'arc dont S est le squelette, A est la surface de la forme squelettisée.

Tab. B.2: Six caractéristiques $(v_1, \ldots, v_6)$ décrivant un arc extrait du squelette d'une forme, elles sont extraites de [Ruberto2004]. L'arc S est défini par ses deux extrémités (x_1, y_1), (x_2, y_2) et une fonction $t \in [0,1] \longrightarrow (x(t), y(t))$.

B.8 Squelette d'un nuage de points

B.8.1 Squelettisation à partir d'un treillis de Kohonen

Le nettoyage des barbules peut s'avérer complexe. De plus, la squelettisation par érosion est souvent sensible aux bruits du contour. C'est pourquoi il est possible de s'inspirer de méthodes qui permettent de déterminer le squelette d'un nuage de points. La figure B.34 montre la construction du squelette de la lettre "A" la méthode développée par [Datta1997] part d'un squelette dont la topologie est linéaire. Elle supprime des neurones si ceux-ci sont trop rapprochés et en ajoute si ceux-ci sont trop éloignés. Elle crée des points "aiguillage" ou barres de "T" lorsque l'angle entre deux segments devient trop fermé.

Soit $(P_1, \ldots, P_N)$ un nuage de points, la topologie initiale est linéaire et le réseau est constitué de n vecteurs (ou neurones) $W = (W_1, \ldots, W_n)$ où W_{i-1} et W_{i+1} sont les voisins du vecteur W_i. A chaque itération t, on tire aléatoirement un point P_i puis on détermine le vecteur $W_{k^*}^t$ qui en est le plus proche à l'itération t :

$$W_{k^*}^t \in \underset{k\in\{1,\ldots,n\}}{\arg\min}\, d\left(W_k^t, P_i\right)$$

$d(W_k, P_i)$ est la distance entre W_k et P_i. On procède ensuite à la mise à jour de W_{k^*} et de l'ensemble de ses voisins noté $N(W_{k^*})$:

$$\forall k \in N(W_{k^*}),\ W_k^{t+1} = W_k^t + \alpha_t [P_i - W_k]$$

La suite $(\alpha_t)_{t\geqslant 0}$ vérifie :

$$\sum_{t\geqslant 0} \alpha_t = \infty \text{ and } \sum_{t\geqslant 0} \alpha_t^2 \leqslant \infty$$

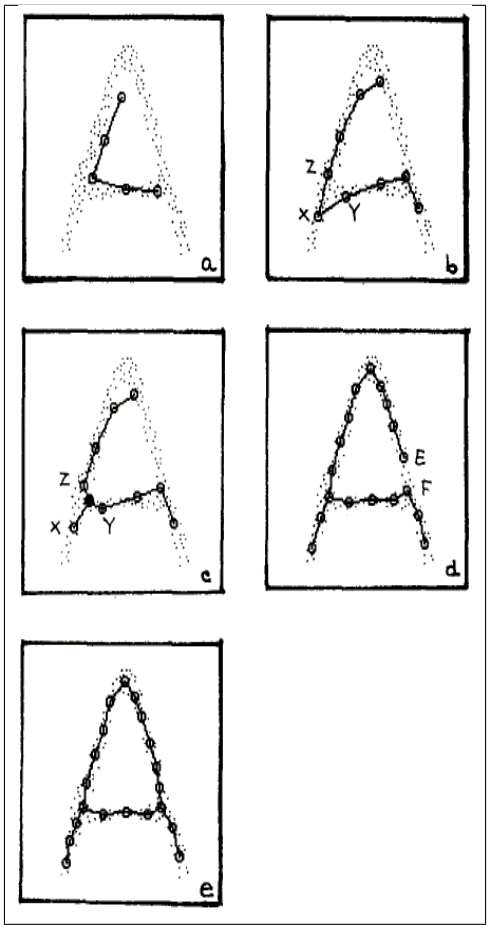


Fig. B.34: Squelette d'un nuage de points : différentes étapes dans la construction du squelette de la lettre "A", figure extraite de [Datta1997].

Par exemple :

$$\alpha_t = \frac{\alpha_0}{1 + \beta t}$$

L'algorithme s'arrête lorsque la condition suivante est vérifiée :

$$\forall i \in \{1, \ldots, n\}\,,\; d\left(W_i^{t+1}, W_i^t\right) \leqslant \epsilon$$

Il reste à gérer la suppression de deux neurones trop proches, l'insertion d'un neurone entre deux autres trop éloignés, l'insertion d'un neurone reliant trois voisins ou neurones "T". Ces opérations sont effectuées une fois que les étapes précédentes ont abouti à une configuration ayant convergé. La suppression d'un neurone est effectuée si la condition suivante est vérifiée :

$$\min_{i \in \{1,\ldots,n\}} \left[\min_{j \in N(W_i)} d\left(W_i, W_j\right) \right] < \delta_1 \tag{B.16}$$

Dans ce cas, les deux neurones permettant d'atteindre le minimum de (B.16) sont regroupés ensemble. L'insertion d'un neurone à deux voisins est effectuée si la condition suivante est vérifiée :

$$\max_{i\in\{1,\ldots,n\}} \left[\max_{j\in N(W_i)} d\left(W_i, W_j\right) \right] > \delta_2 \tag{B.17}$$

Dans ce cas, un neurone est insérée au milieu du segment formé par les deux voisins permettant d'obtenir le maximum de (B.17). Un neurone "T" à trois voisins est inséré lorsque l'angle entre les deux voisins d'un neurone forme un angle fermé. La figure B.34*(b)* montre trois points X,Y,Z. Les droites (XZ) et (XY) forment un angle fermé, un point à trois voisins est alors ajouté entre les points Y et Z.

B.8.2 Squelettisation à partir d'un treillis de Kohonen

Cette segmentation s'inspire de l'article [Datta1997] (voir aussi [Singh2000]). Cet article permet de construire le squelette d'un nuage de points. L'indépendance par rapport à la connexité permet d'être moins sensible au bruit. Afin de simplifier l'algorithme décrit dans [Datta1997], un treillis en deux dimensions est d'abord superposé à l'image du mot ainsi que le montre la figure B.35.

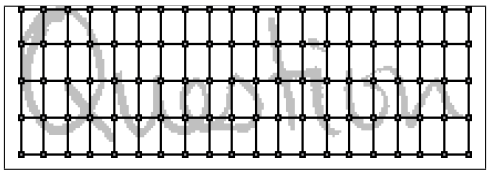


Fig. B.35: Treillis de Kohonen initial superposé au mot à segmenter en graphèmes.

Le treillis évolue puis converge vers le résultat figure B.36*a* en utilisant l'algorithme B.8.1 [2] déjà décrit au paragraphe 4.2.3 (page 82). Le résultat obtenu inclut le squelette recherché qui sera obtenu en utilisant un algorithme de recherche de l'arbre de poids minimal (voir [Kruskal1956]).

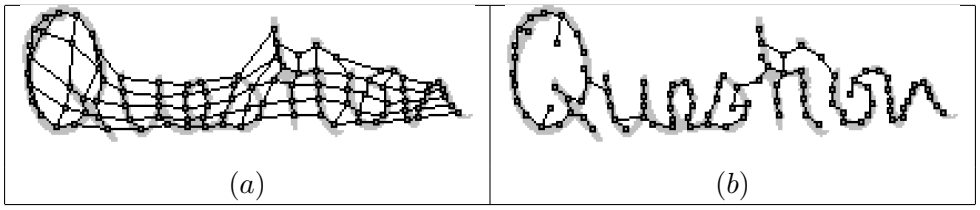
Algorithme B.8.1 : caractéristiques de Kohonen
Etape A : initialisation
$\forall\,(i,j) \in \{1,\ldots,I\} \times \{1,\ldots,J\}\,,\ P_{ij} = \left(\frac{iX}{I}, \frac{jY}{J}\right)'$
$t \leftarrow 0$
$\delta \leftarrow \sqrt{\left(\frac{X}{I}\right)^2 + \left(\frac{Y}{J}\right)^2}$

Etape B : point caractéristique le plus proche et mise à jour
$\alpha \leftarrow \frac{0.2}{1+\frac{t}{XY}}$
On tire aléatoirement un pixel p de l'image, si ce pixel p est noir, alors :
$(i^*, j^*) \leftarrow \arg\max_{i,j} d\left(P_{ij}, p\right)$
$P_{i^*,j^*} \leftarrow P_{i^*,j^*} + \alpha\left(p - P_{i^*,j^*}\right)$

Etape C : mise à jour des voisins
$\epsilon \leftarrow \exp\left(\frac{1}{\delta}\left\|P_{i^*,j^*} - p\right\| - 1\right)$
pour chaque $P \in V_c\left(i^*, j^*\right)$ **faire**
  $\beta \leftarrow \alpha\,\epsilon\,\exp\left(-\frac{1}{\delta}\left\|P - p\right\|\right)$
  $P \leftarrow P + \beta\left(p - P\right)$
fin pour

Etape D : terminaison
$t \leftarrow t + 1$
Tant que l'algorithme n'a pas convergé, retour à l'étape B.

2. Pour mémoire :



(a) (b)

Fig. B.36: La figure (*a*) montre le résultat obtenu après convergence de l'algorithme 4.2.3. La figure (*b*) montre le même arbre élagué après l'application de l'algorithme de Kruskal.

B.8.3 Squelettisation d'images floues

L'article [Kalmár1999] propose une squelettisation d'images floues basée sur un réseau de neurones. Sa méthode propose l'érosion d'une image en niveaux de gris. On suppose que l'image est formée d'une suite de pixels $(x_i)_i$ et $f(x_i) \in [0,1]$ est l'intensité du pixel. Les neurones de la première couche sont notés $\left(n_i^1\right)_i$, deux points p_i, $a_i \in \mathbb{R}^2$ sont associés à chacun de ces neurones, (a_i) est calculé comme suit :

$$\forall i,\ a_i = \sum_j x_j\ f(x_j)\ \exp\left[\frac{-\|p_i - x_j\|^2}{d_0^2}\right] \tag{B.18}$$

d_0 est un paramètre d'échelle. Les points p_i sont éparpillés sur l'image et connectés entre eux selon un système de voisinage qui peut être aussi bien carré (4, 8-connexité) qu'hexagonal. La connexion entre les neurones n_i et n_j est notée $w_{ij} \in [0,1]$. A chaque neurone est associé un paramètre d'activation qui dépend du temps $\sigma_i(t) \in [0,1]$. $\sigma_i(0)$ est estimé à partir de a_i. Par exemple :

$$\sigma_i(0) = f(a_i) \tag{B.19}$$

L'évolution de $\sigma_i(t)$ est définie comme suit :

$$\frac{\partial \sigma_i}{\partial t}(t) = (1 - \sigma_i(t)) \left(\sum_{k,j} w_{ij}\left[\sigma_k(t) - \sigma_i(t)\right]\right) \tag{B.20}$$

On définit le coefficient $\Sigma_i(t)$ pour chaque neurone :

$$\Sigma_i(t) = \int_0^t \sum_k \left[\sigma_k(u) - \sigma_i(u)\right] du \tag{B.21}$$

A partir d'un certain temps τ, la fonction $\Sigma_i(t)$ converge pour tous les neurones et la suite $(\Sigma_i(\tau))_i$ est alors la distribution du squelette sur l'ensemble des neurones. Cette méthode peut également être appliquée sur des images binaires mais le résultat n'est plus un squelette d'un pixel d'épaisseur un (ou "fil de fer"). En contrepartie, les branches non significatives sont moins probables (voir figure B.37).

B.8.4 Squelettisation et classification

La squelettisation peut être également traitée comme une classification non supervisée. L'article [Liu2003] propose une méthode [3] qui considère chaque segment de l'image comme une classe suivant une loi normale

3. Annexes : voir paragraphe H.3.3, page 356

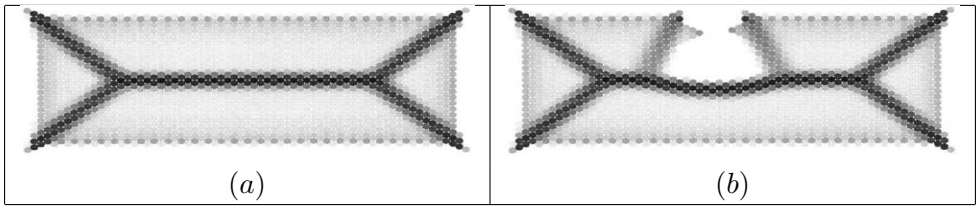


Fig. B.37: Figure extraite de [Kalmár1999], la forme (b) est amputée d'un bout dans sa moitié supérieure, les artefacts qui en résultent au niveau du squelette sont moins probables que les autres.

multidimensionnelle et dégénérée, la forme de cette classe représente une ellipse dont le petit axe est très inférieur au grand axe. L'algorithme RPCL-local based PCA permet à la fois de déterminer le nombre de segments et d'estimer les paramètres de la loi qui le modélise. L'article [ZhangB2004] présente un nouvel algorithme EM appelé Competitive EM [4] permettant d'éviter les maxima locaux lors de l'optimisation. Le résultat obtenu sur des caractères chinois est présenté par la figure B.38.

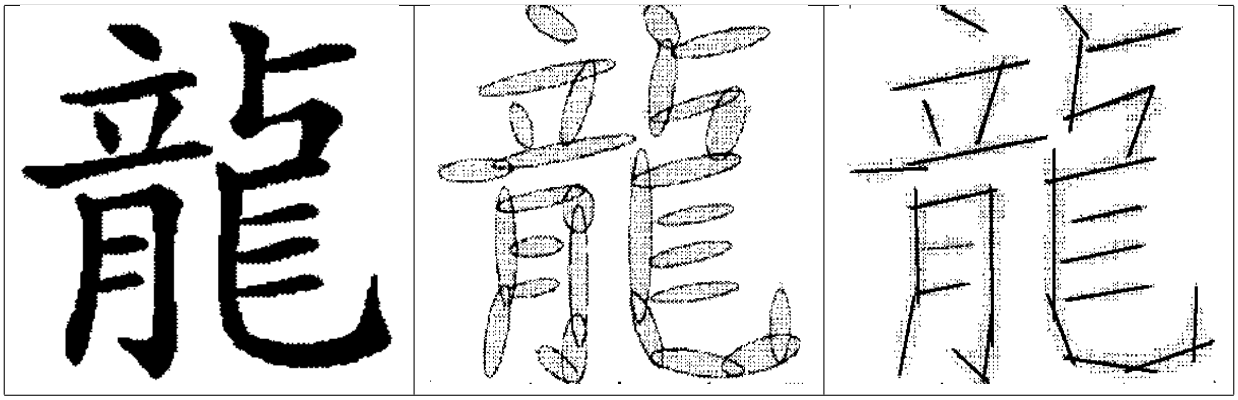


Fig. B.38: Figures extraites de [ZhangB2004] présentant les résultats d'une classification effectuée par l'algorithme Competitive Expectation Maximization (CEM). La première image représente le caractère chinois à segmenter. La seconde image illustre les classes obtenues après que l'image a été échantillonnée (environ 1500 points). La dernière traduit chaque classe par un segment égal au grand axe de l'ellipse.

4. Annexes : voir paragraphe H.4.2, page 360

Annexe C

Réseaux de neurones

Ce chapitre aborde les réseaux de neurones au travers de deux utilisations courantes, la régression (paragraphe C.2) et la classification (paragraphe C.5), et une qui l'est moins, l'analyse en composantes principales ou ACP (paragraphe C.8), sans oublier les méthodes d'estimation des paramètres qui les composent, à savoir optimisations du premier et second ordre (paragraphes C.4.1.1 et C.4.1.2) ainsi qu'une méthode permettant de supprimer des coefficients inutiles (paragraphe C.7).

C.1 Définition des réseaux de neurones multi-couches

Les réseaux de neurones multi-couches (ou perceptrons) définissent une classe de fonctions dont l'intérêt est de pouvoir approcher n'importe quelle fonction continue à support compact (voir théorème C.2.1 page 199). Aucun autre type de réseau de neurones ne sera étudié et par la suite, tout réseau de neurones sera considéré comme multi-couches.

C.1.1 Un neurone

Définition C.1.1 : neurone
Un neurone à p entrées est une fonction $f : \mathbb{R}^{p+1} \times \mathbb{R}^p \longrightarrow \mathbb{R}$ définie par :

1. $g : \mathbb{R} \longrightarrow \mathbb{R}$
2. $W \in \mathbb{R}^{p+1}$, $W = (w_1, \ldots, w_{p+1})$
3. $\forall x \in \mathbb{R}^p,\ f(W, x) = g\left(\sum_{i=1}^{p} w_i x_i + w_{p+1}\right)$
   avec $x = (x_1, \ldots, x_p)$

Cette définition est inspirée du neurone biologique, les poids jouant le rôle de synapses, le vecteur x celui des *entrées* et W celui des *coefficients* ou *poids*. Le coefficient w_{p+1} est appelé le *biais* et souvent noté b. La fonction g est appelée *fonction de transfert* ou *fonction de seuil*. A cette définition mathématique du neurone, une autre moins formelle et plus graphique (figure C.1) lui est préférée. Ce schéma est également plus proche de sa définition biologique et dissocie mieux les rôles non symétriques des entrées et des poids. Des exemples de fonctions de transfert sont donnés par la table C.1 dont les plus couramment utilisées sont les fonctions linéaire et sigmoïde.

La plupart des fonctions utilisées sont dérivables et cette propriété s'étend à tout assemblage de neurones, ce qui permet d'utiliser l'algorithme de rétropropagation découvert par [Rumelhart1986]. Ce dernier permet le calcul de la dérivée ouvre ainsi les portes des méthodes d'optimisation basées sur cette propriété.

Le vecteur $(x_1, ..., x_p) \in \mathbb{R}^p$ joue le rôle des *entrées*.
y est appelé parfois le *potentiel*.

$$y = \sum_{i=1}^{p} w_i x_i + b$$

z est appelée la sortie du neurone.
f est appelée la fonction de transfert ou de seuil.

$$z = f(y) = f\left(\sum_{i=1}^{p} w_i x_i + b\right)$$

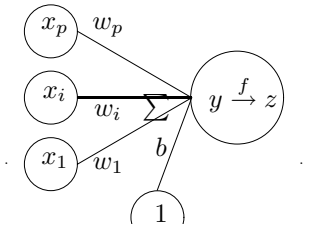


Fig. C.1: Un neurone.

exemples de fonction de transfert ou de seuil	**expression : $\mathbf{f}(\mathbf{x}) =$**
escalier	$1_{[0,+\infty[}$
linéaire	x
sigmoïde entre $[0,1]$	$\dfrac{1}{1+e^{-x}}$
sigmoïde entre $[-1,1]$	$1 - \dfrac{2}{1+e^{x}}$
normale	$e^{-\frac{x^2}{2}}$
exponentielle	e^{x}

Tab. C.1: Fonctions de transfert usuelles.

C.1.2 Une couche de neurones

Définition C.1.2 : couche de neurones
Soit p et n deux entiers naturels, on note $W \in \mathbb{R}^{n(p+1)} = (W_1, \ldots, W_n)$
avec $\forall i \in \{1, \cdots, n\},\ W_i \in \mathbb{R}^{p+1}$.
Une couche de n neurones et p entrées est une fonction :

$$F : \mathbb{R}^{n(p+1)} \times \mathbb{R}^p \longrightarrow \mathbb{R}^n$$

vérifiant :

1. $\forall i \in \{1, \cdots, n\},\ f_i$ est un neurone.
2. $\forall W \in \mathbb{R}^{n(p+1)} \times \mathbb{R}^p,\ F(W, x) = (f_1(W_1, x), \ldots, f_n(W_n, x))$

Une couche de neurones représente la juxtaposition de plusieurs neurones partageant les mêmes entrées mais ayant chacun leur propre vecteur de coefficients et leur propre sortie.

C.1.3 Un réseau de neurones : le perceptron

Définition C.1.3 : réseau de neurones multi-couches ou perceptron (figure C.2)
Un réseau de neurones multi-couches à n sorties, p entrées et C couches est une liste de couches $(C_1, \ldots, C_C)$ connectées les unes aux autres de telle sorte que :

1. $\forall i \in \{1, \cdots, C\}$, chaque couche C_i possède n_i neurones et p_i entrées
2. $\forall i \in \{1, \cdots, C-1\}$, $n_i = p_{i+1}$, de plus $p_1 = p$ et $n_C = n$
3. les coefficients de la couche C_i sont notés $\left(W_1^i, \ldots, W_{n_i}^i\right)$, cette couche définit une fonction F_i
4. soit la suite $(Z_i)_{0 \leqslant i \leqslant C}$ définie par :

$$Z_0 \in \mathbb{R}^p$$
$$\forall i \in \{1, \cdots, C\}, \; Z_i = F_i\left(W_1^i, \ldots, W_{n_i}^i, Z_{i-1}\right)$$

On pose $M = M = \sum_{i=1}^{C} n_i \left(p_i + 1\right)$, le réseau de neurones ainsi défini est une fonction F telle que :

$$\begin{array}{rccc} F: & \mathbb{R}^M \times \mathbb{R}^p & \longrightarrow & \mathbb{R}^n \\ & (W, Z_0) & \longrightarrow & Z_C \end{array}$$

$(x_1, \ldots, x_p)$	entrées
C_i	nombre de neurones sur la couche i ($C_0 = p$)
$z_{c,i}$	sortie du neurone i de la couche c par extension, $z_{0,i} = x_i$
$y_{c,i}$	potentiel du neurone i de la couche c
$w_{c,i,j}$	coefficient associé à l'entrée j du neurone i de la couche c
$b_{c,i}$	biais du neurone i de la couche c
$f_{c,i}$	fonction de seuil du neurone i de la couche c

alors $\begin{cases} y_{c,i} = \sum_{j=1}^{C_{c-1}} w_{c,i,j} z_{c-1,j} + b_{c,i} \\ z_{c,i} = f\left(y_{c,i}\right) \end{cases}$

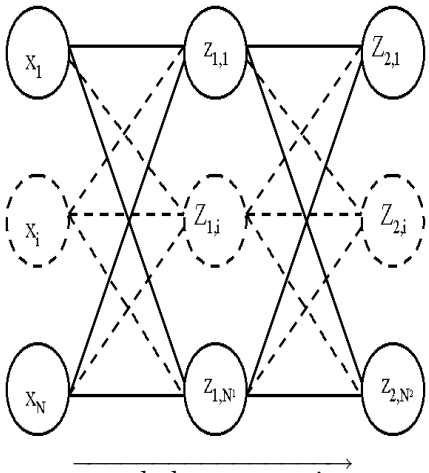


Fig. C.2: Modèle du perceptron multi-couche (multi-layer perceptron, MLP).

Souvent, on considère que les entrées forment la couche C_0 de manière à simplifier les écritures. Ainsi, chaque couche C_i du perceptron a pour entrées les sorties de la couche C_{i-1}. Cette définition est plus

facile à illustrer qu'à énoncer (figure C.2) et rappelle le rôle non symétrique des entrées et des poids. Le mécanisme qui permet de calculer les sorties d'un réseau de neurones sachant ses poids est appelé *propagation* :

Algorithme C.1.4 : propagation
Cet algorithme s'applique à un réseau de neurones vérifiant la définition C.1.3. Il s'agit de calculer les sorties de ce réseau connaissant ses poids $(w_{c,i,j})$ et ses entrées (x_j).

Etape A : initialisation
pour $i = 1$ **à** C_0 **faire**
$z_{0,i} \longleftarrow x_i$
fin pour

Vient ensuite le calcul itératif de la suite $(Z_c)_{1 \leqslant c \leqslant C}$:

Etape B : récurrence
pour $c = 1$ **à** C **faire**
pour $i = 1$ **à** C_c **faire**
$z_{c,i} \longleftarrow 0$
pour $j = 1$ **à** C_{i-1} **faire**
$z_{c,i} \longleftarrow z_{c,i} + w_{c,i,j} z_{c-1,j}$
fin pour
$z_{c,i} \longleftarrow f\left(z_{c,i} + b_{c,i}\right)$
fin pour
fin pour

Le nombre de couches d'un réseau de neurones n'est pas limité. Les réseaux de deux couches (une couche pour les entrées, une couche de sortie) sont rarement utilisés. Trois couches sont nécessaires (une couche pour les entrées, une couche dite *cachée*, une couche de sortie) pour construire des modèles avec une propriété intéressante de densité (théorème C.2.1).

C.1.4 La régression

Le bruit blanc est une variable aléatoire couramment utilisé pour désigner le hasard ou la part qui ne peut être modélisée dans une régression ou tout autre problème d'apprentissage. On suppose parfois que ce bruit suive une loi normale.

Définition C.1.5 : bruit blanc gaussien
Une suite de variables aléatoires réelles $(\epsilon_i)_{1 \leqslant i \leqslant N}$ est un bruit blanc gaussien :

1. $\exists \sigma > 0, \forall i \in \{1, \cdots, N\}\,,\ \epsilon_i \sim \mathcal{N}(0, \sigma)$
2. $\forall (i,j) \in \{1, \cdots, N\}^2\,,\ i \neq j \Longrightarrow \epsilon_i \perp\!\!\!\perp \epsilon_j$

Une régression consiste à résoudre le problème suivant :

Problème C.1.6 : régression
Soient deux variables aléatoires X et Y, l'objectif est d'approximer la fonction $E(Y \mid X) = f(X)$.
Les données du problème sont :
- un échantillon de points : $\{(X_i, Y_i) \,|1 \leqslant i \leqslant N\}$
- un modèle :

$$\text{soit } \theta \in \mathbb{R}^n,\ \forall i \in \{1, \cdots, N\},\ Y_i = f(\theta, X_i) + \epsilon_i$$

avec :
- $n \in \mathbb{N}$
- $(\epsilon_i)_{1 \leqslant i \leqslant N}$ bruit blanc (voir définition C.1.5)
- f est une fonction de paramètre θ

La fonction f peut être :

- une fonction linéaire
- un polynôme
- un réseau de neurones
- ...

Lorsque le bruit blanc est normal, la théorie de l'estimateur de vraisemblance ([Saporta1990]) permet d'affirmer que le meilleur paramètre $\hat{\theta}$ minimisant l'erreur de prédiction est :

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^p}{\arg\min}\ E(\theta) = \underset{\theta \in \mathbb{R}^p}{\arg\min} \left[\sum_{i=1}^{N} [Y_i - f(\theta, X_i)]^2 \right]$$

Le lien entre les variables X et Y dépend des hypothèses faites sur f. Généralement, cette fonction n'est supposée non linéaire que lorsqu'une régression linéaire donne de mauvais résultats (figure C.3). Cette hypothèse est toujours testée car la résolution du problème dans ce cas-là est déterministe et aboutit à la résolution d'un système linéaire avec autant d'équations que d'inconnues.

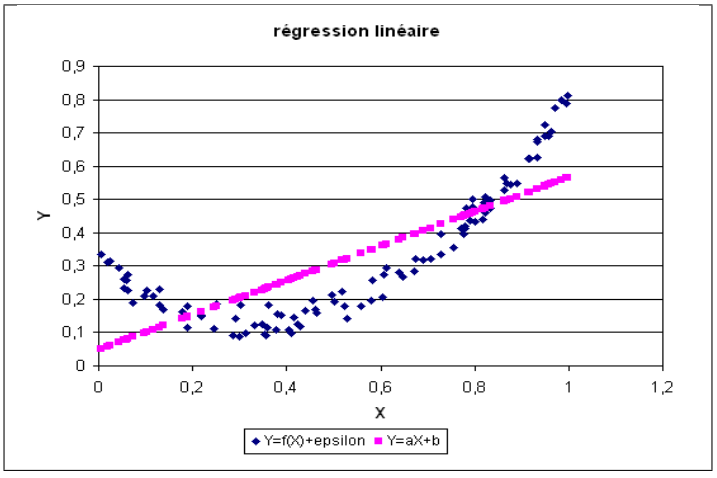


Fig. C.3: Régression linéaire de $Y = \frac{3}{2}X^2 - X + \frac{1}{4} + \epsilon$

Le schéma C.4 illustre une régression non linéaire $Y = f_1(X) + \epsilon$ où la fonction f_1 est un réseau de neurones :

- il n'y a qu'une entrée

– l'unique couche cachée ne contient qu'un neurone dont la fonction de transfert est sigmoïde
– la couche de sortie est linéaire
– $\theta \in \mathbb{R}^n$ représente le vecteur des poids du réseau de neurones, ici $n = 4$

Le schéma C.5 illustre la même régression avec deux neurones sur la couche cachée, ou avec cent neurones sur la couche cachée, figure C.6.

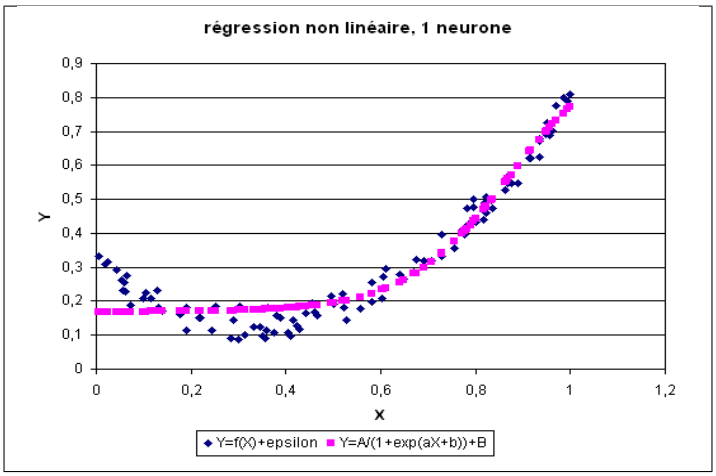


Fig. C.4: Régression non linéaire (1 neurone sur la couche cachée) de $Y = \frac{3}{2}X^2 - X + \frac{1}{4} + \epsilon$

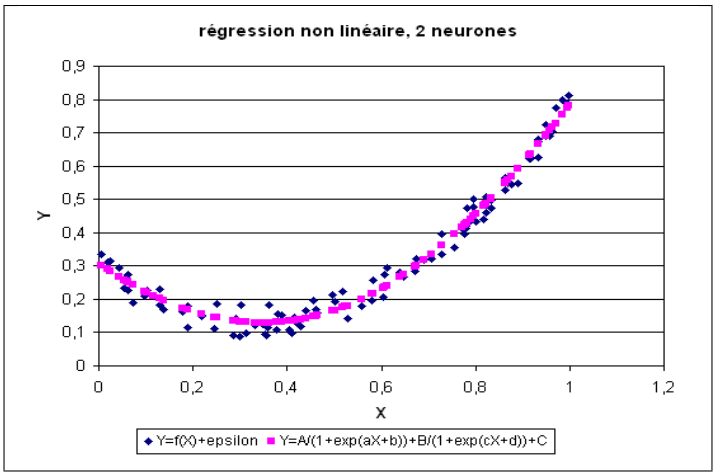


Fig. C.5: Régression non linéaire (2 neurones sur la couche cachée) de $Y = \frac{3}{2}X^2 - X + \frac{1}{4} + \epsilon$

Dans le cas d'une régression à cent neurones, le nombre de coefficients du réseau de neurones (301) est largement supérieur au nombre de points (50). Il en résulte que contrairement aux trois précédents cas, la "richesse" du modèle choisi lui permet d'apprendre le "hasard". Lorsque ce cas de figure se présente, on dit que le réseau de neurones a appris *par cœur* et que son *pouvoir de généralisation* est mauvais : l'erreur minime estimée sur ce nuage de points (ou *base d'apprentissage*) sera considérablement accrue sur un autre nuage de points (ou *base de test*) suivant la même loi.

Cet exemple montre que le choix du réseau de neurones le mieux adapté au problème n'est pas évident. Il existe des méthodes permettant d'approcher l'architecture optimale mais elles sont généralement coûteuses en calcul (paragraphe C.7).

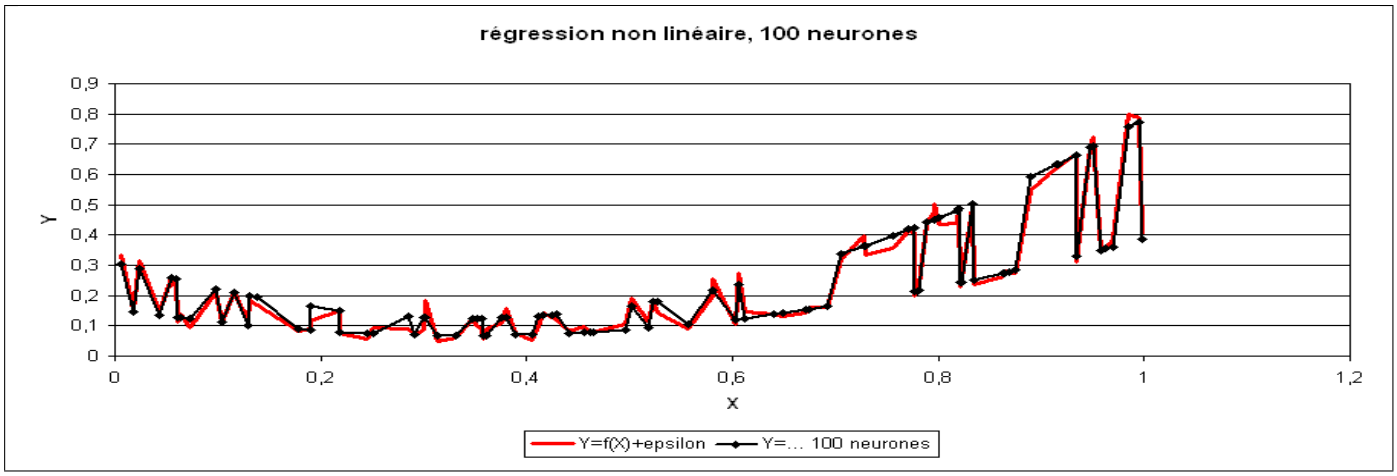


Fig. C.6: Régression non linéaire (100 neurones sur la couche cachée) de $Y = \frac{3}{2}X^2 - X + \frac{1}{4} + \epsilon$, l'erreur de prédiction de ce réseau de neurones est très inférieure à celle des modèles des figures C.3, C.4, C.5, ce modèle a appris par cœur le nuage de points (X_i, Y_i) sans vraiment "comprendre" ce qu'il apprenait.

C.1.5 La classification

Comme la régression, la classification consiste aussi à trouver le lien entre une variable X et une variable aléatoire discrète suivant une loi multinomiale Y.

Problème C.1.7 : classification
Soit une variable aléatoire X et une variable aléatoire discrète Y, l'objectif est d'approximer la fonction $E(Y \mid X) = f(X)$.
Les données du problème sont :
- un échantillon de points : $\{(X_i, Y_i) \,|1 \leqslant i \leqslant N\}$ avec $\forall i \in \{1, \ldots, N\}$, $Y_i \in \{1, \ldots, C\}$
- un modèle :

$$\text{soit } \theta \in \mathbb{R}^n,\ \forall i \in \{1, \cdots, N\},\ \forall c \in \{1, \cdots, C\},\ \mathbb{P}(Y_i = c \mid X_i, \theta) = h(\theta, X_i, c)$$

avec :
- $n \in \mathbb{N}$
- h est une fonction de paramètre θ à valeur dans $[0, 1]$ et vérifiant la contrainte : $\sum_{c=1}^{C} h(\theta, X, c) = 1$

L'exemple C.7 est une classification en deux classes, elle consiste à découvrir le lien qui unit une variable aléatoire réelle X et une variable aléatoire discrète et $Y \in \{0, 1\}$, on dispose pour cela d'une liste :

$$\{(X_i, Y_i) \in \mathbb{R} \times \{0, 1\} \mid 1 \leqslant i \leqslant N\}$$

Il n'est pas facile de déterminer directement une fonction h qui approxime $Y \mid X$ car h et Y sont toutes deux discrètes. C'est pourquoi, plutôt que de résoudre directement ce problème, il est préférable de déterminer la loi marginale $\mathbb{P}(Y = c|X) = f(X, \theta, c)$. f est alors une fonction dont les sorties sont continues et peut être choisie dérivable. Par exemple, f peut être un réseau de neurones dont les sorties vérifient :

$$f(X, 0) + f(X, 1) = \mathbb{P}(0|X) + \mathbb{P}(1|X) = 1$$

Le réseau de neurones utilisé pour cette tâche est légèrement différent du précédent, il sera présenté ultérieurement dans le paragraphe C.5, page 217.

Dans le schéma C.7, un plan a été divisé en deux demi-plan par une droite délimitant deux classes, le réseau de neurones dont la couche cachée contient deux neurones linéaires, a retrouvé cette séparation malgré les quelques exemples mal classés. En revanche, un réseau de neurones comportant trop de coefficients aura tendance à apprendre par cœur la classification et les quelques erreurs de classification comme le montre la figure C.8. La séparation produite par le réseau de neurones est de manière évidente non linéaire puisqu'aucune droite ne peut séparer les deux classes déterminées par cette fonction. Cette classe de modèles permet donc de résoudre des problèmes complexes en gardant toutefois à l'esprit, comme dans le cas de la régression, qu'il n'est pas moins de facile de dénicher le bon modèle que dans le cas linéaire.

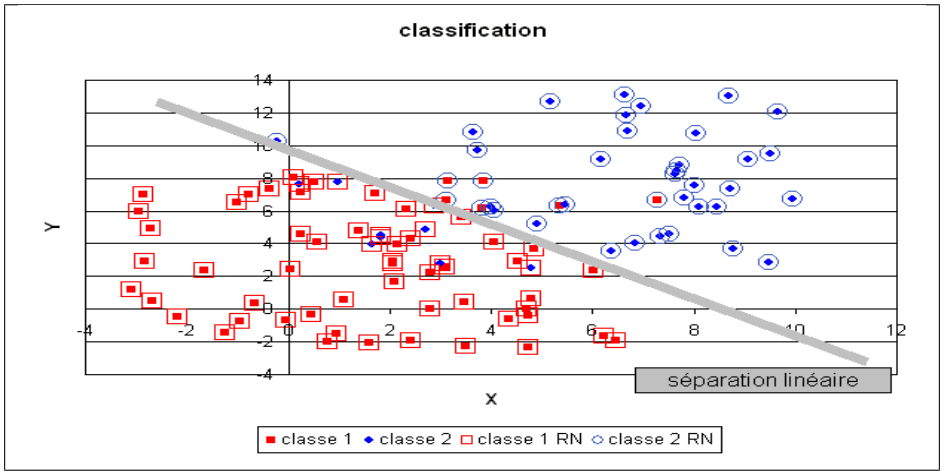


Fig. C.7: Classification d'un plan en deux demi-plans.

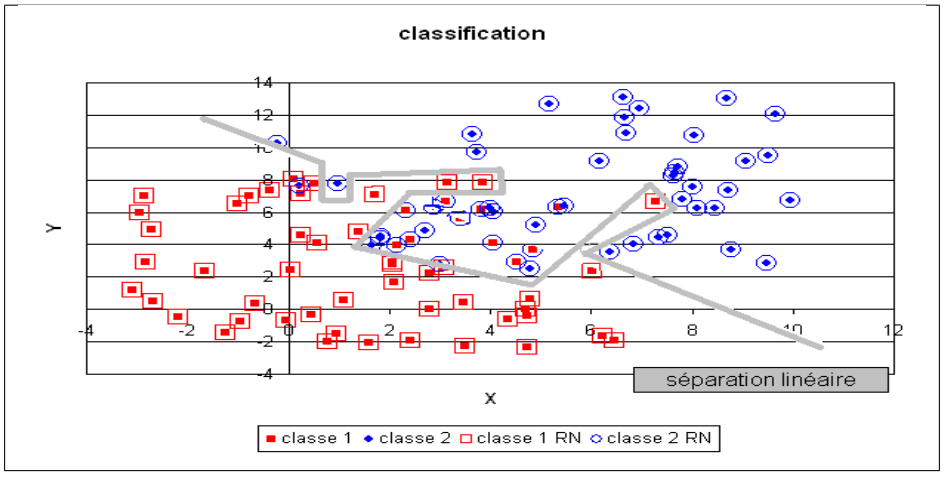


Fig. C.8: Classification rendue par un réseau de 30 neurones.

C.2 Régression par un réseau de neurones multi-couches

C.2.1 Résolution du problème de la régression

Soient deux variables aléatoires continues $(X, Y) \in \mathbb{R}^p \times \mathbb{R}^q \sim \mathcal{L}$ quelconque, la résolution du problème C.1.6 est l'estimation de la fonction :

$$\mathbb{E}(Y|X) = F(X) \tag{C.1}$$

Pour cela, on dispose d'un ensemble de points :

$$A = \{(X_i, Y_i) \sim \mathcal{L} | 1 \leqslant i \leqslant N\}$$

Soit $f : \mathbb{R}^M \times \mathbb{R}^p \longrightarrow \mathbb{R}^q$ une fonction, on définit :

$$\forall i \in \{1, \cdots, N\}, \ \widehat{Y_i^W} = f(W, X_i)$$

$\widehat{Y_i^W}$ est appelée la valeur prédite pour X_i. On pose alors :

$$\forall i \in \{1, \cdots, N\}, \ \epsilon_i^W = Y_i - \widehat{Y_i^W} = Y_i - f(W, X_i)$$

Les résidus sont supposés i.i.d. (identiquement et indépendemment distribués), et suivant une loi normale :

$$\forall i \in \{1, \cdots, N\}, \ \epsilon_i^W \sim \mathcal{N}(\mu_W, \sigma_W)$$

La log-vraisemblance [1] de l'échantillon est alors :

$$L_W = -\frac{1}{2\sigma_W^2} \sum_{i=1}^{N} \left(Y_i - \widehat{Y_i^W} - \mu_W\right)^2 + N \ln\left(\sigma_W \sqrt{2\pi}\right)$$

Les estimateurs du maximum de vraisemblance pour μ_W et σ_W sont (voir [Saporta1990]) :

$$\widehat{\mu_W} = \frac{1}{N} \sum_{i=1}^{N} Y_i - \widehat{Y_i^W} \quad \text{et} \quad \widehat{\sigma_W} = \sqrt{\frac{\sum_{i=1}^{N} \left(Y_i - \widehat{Y_i^W} - \mu_W\right)^2}{N}}$$

L'estimateur de $\widehat{Y} = f(W, X)$ désirée est de préférence sans biais ($\mu_W = 0$) et de variance minimum, par conséquent, les paramètres $\overset{*}{W}$ qui maximisent la vraisemblance L_W sont :

1. La vraisemblance d'un échantillon $(Z_i)_{1 \leqslant i \leqslant N}$, où les Z_i sont indépendantes entre elles et suivent la loi de densité $f(z|\theta)$ est la densité du vecteur $(Z_1, \ldots, Z_N)$:

$$L(\theta, Z_1, \ldots, Z_N) = \prod_{n=1}^{N} f(Z_i|\theta) \Longrightarrow \ln L(\theta, Z_1, \ldots, Z_N) = \sum_{n=1}^{N} \ln f(Z_i|\theta)$$

$$\overset{*}{W} = \underset{W\in\mathbb{R}^M}{\arg\min} \sum_{i=1}^{N} \left(Y_i - \widehat{Y_i^W}\right)^2 = \underset{W\in\mathbb{R}^M}{\arg\min} \sum_{i=1}^{N} \left(Y_i - f\left(W, X_i\right)\right)^2 \tag{C.2}$$

Réciproquement, on vérifie que si W^* vérifie l'équation (C.2) alors l'estimateur défini par f est sans biais [2] et minimise la vraisemblance L_W. Cette formule peut être généralisée en faisant une autre hypothèse que celle de la normalité des résidus (l'indépendance étant conservée), l'équation (C.2) peut généralisée par (C.3) :

$$\overset{*}{W} = \underset{W\in\mathbb{R}^M}{\arg\min} \sum_{i=1}^{N} e\left(Y_i - \widehat{Y_i^W}\right) = \underset{W\in\mathbb{R}^M}{\arg\min} \sum_{i=1}^{N} e\left(Y_i - f\left(W, X_i\right)\right) \tag{C.3}$$

où la fonction $e : \mathbb{R}^q \longrightarrow \mathbb{R}$ est appelée fonction d'erreur

C.2.2 Propriété et intérêt des réseaux de neurones

L'utilisation de réseaux de neurones s'est considérablement développée depuis que l'algorithme de rétro-propagation a été trouvé ([LeCun1985], [Rumelhart1986], [Bishop1995]). Ce dernier permet d'estimer la dérivée d'un réseau de neurones en un point donné et a ouvert la voie à des méthodes classiques de résolution pour des problèmes d'optimisation tels que la régression non linéaire.

Comme l'ensemble des fonctions polynômiales, l'ensemble des fonctions engendrées par des réseaux de neurones multi-couches possède des propriétés de densité (théorème C.2.1) et sont infiniment dérivables. Les réseaux de neurones comme les polynômes sont utilisés pour modéliser la fonction f de l'équation C.3. Ils diffèrent néanmoins sur certains points

Si une couche ne contient que des fonctions de transfert bornées comme la fonction sigmoïde, tout réseau de neurones incluant cette couche sera aussi borné. D'un point de vue informatique, il est préférable d'effectuer des calculs avec des valeurs du même ordre de grandeur. Pour un polynôme, les valeurs des termes de degré élevé peuvent être largement supérieurs à leur somme.

Un autre attrait est la symétrie dans l'architecture d'un réseau de neurones, les neurones qui le composent jouent des rôles symétriques (corollaire C.2.4). Pour améliorer l'approximation d'une fonction, dans un cas, il suffit d'ajouter un neurone au réseau, dans l'autre, il faut inclure des polynômes de degré plus élevé que ceux déjà employés.

Théorème C.2.1 : densité des réseaux de neurones (Cybenko1989)
(voir [Cybenko1989])
Soit E_p^q l'espace des réseaux de neurones à p entrées et q sorties, possédant une couche cachée dont la fonction de seuil est une fonction sigmoïde $\left(x \to 1 - \frac{2}{1+e^x}\right)$, une couche de sortie dont la fonction de seuil est linéaire (voir figure C.15, page 228).
Soit F_p^q l'ensemble des fonctions continues de $C \subset \mathbb{R}^p \longrightarrow \mathbb{R}^q$ avec C compact muni de la norme :

$$\|f\| = \sup_{x\in C} \|f\left(x\right)\|$$

Alors E_p^q est dense dans F_p^q.

2. Il suffit pour s'en convaincre de poser $g = f + \alpha$ avec $\alpha \in \mathbb{R}$ et de vérifier que la valeur optimale pour α est $\alpha = -\frac{1}{N}\sum_{i=1}^{N} Y_i - f\left(W, X_i\right)$.

La démonstration de ce théorème nécessite deux lemmes. Ceux-ci utilisent la définition usuelle du produit scalaire [3] et la norme infinie [4].

Lemme C.2.2 : approximation d'une fonction créneau
Soit $C \subset \mathbb{R}^p$, $C = \{(y_1, \ldots, y_p) \in \mathbb{R}^p \mid \forall i \in \{1, \cdots, p\}, 0 \leqslant y_i \leqslant 1\}$, alors :

$$\forall \varepsilon > 0,\ \forall \alpha > 0,\ \exists n \in \mathbb{N}^*,\ \exists (x_1, \ldots, x_n) \in (\mathbb{R}^p)^n,\ \exists (\gamma_1, \ldots, \gamma_n) \in \mathbb{R}^n \text{ tels que } \forall x \in \mathbb{R}^p,$$

$$\left| \sum_{i=1}^{n} \frac{\gamma_i}{1 + e^{\langle x_i, x \rangle + b_i}} - \mathbb{1}_{\{x \in C\}} \right| \leqslant 1$$

$$\text{et} \quad \inf_{y \in Fr(C)} \|x - y\| > \alpha \Rightarrow \left| \sum_{i=1}^{n} \frac{\gamma_i}{1 + e^{\langle x_i, x \rangle + b_i}} - \mathbb{1}_{\{x \in C\}} \right| \leqslant \varepsilon$$

Démonstration (lemme C.2.2) :

Partie A (démonstration de C.2.2)

Soit h la fonction définie par : $h(x) = \left(\frac{1}{1 + e^{-kx}} \right)^p$ avec $p > 0$ et $0 < \epsilon < 1$.

A α, ϵ fixé, $0 < \epsilon < 1$, on cherche k tel que :

$$\begin{aligned}
\epsilon &= h(\alpha) = \left(\frac{1}{1 + e^{-k\alpha}} \right)^p \\
\Longrightarrow \quad \epsilon^{-\frac{1}{p}} &= 1 + e^{-k\alpha} \\
\Longrightarrow \quad \epsilon^{-\frac{1}{p}} - 1 &= e^{-k\alpha} \\
\Longrightarrow \quad \ln\left(\epsilon^{-\frac{1}{p}} - 1\right) &= -k\alpha \\
\Longrightarrow \quad k &= -\frac{\ln\left(\epsilon^{-\frac{1}{p}} - 1\right)}{\alpha} = k_0(\epsilon, \alpha, p)
\end{aligned}$$

Partie B (démonstration de C.2.2)

Soit $\alpha > 0$ et $1 \geqslant \varepsilon > 0$, $k > 0$,

On pose $f(y_1, ..., y_p) = \prod_{i=1}^{p} \frac{1}{1 + e^{-ky_i}} \prod_{i=1}^{p} \frac{1}{1 + e^{-k(1-y_i)}}$, d'après sa définition, $0 \leqslant f(y_1, ..., y_p) \leqslant 1$

Pour $k \geqslant k_0(\epsilon, \alpha, 2p)$ obtenu dans la partie A, on a :

$$\inf_{i \in \{1, \ldots, p\}} [\min\{|y_i|, |1 - y_i|\}] > \alpha \Longrightarrow \left\| f(y_1, ..., y_p) - \mathbb{1}_{\{x \in C\}} \right\| \leqslant \varepsilon$$

3. Le produit scalaire sur $\mathbb{R}^p$ est défini par :

$$(x, y) = (x_1, \ldots, x_p, y_1, \ldots, y_p) \in \mathbb{R}^{2p} \longrightarrow \langle x, y \rangle = \sum_{i=1}^{p} x_i y_i$$

4. Comme toutes les normes sont équivalentes sur $\mathbb{R}^p$, on définit la norme :

$$x = (x_1, \ldots, x_p) \in \mathbb{R}^p \longrightarrow \|x\| = \max_{i \in \{1, \cdots, p\}} x_i$$

Partie C (démonstration de C.2.2)

Soit g la fonction définie par :

$$g(x) = \left(\frac{1}{1+e^{-kx}}\right)\left(\frac{1}{1+e^{-k(1-x)}}\right) = \frac{1}{1+e^{-kx}+e^{-k(1-x)}+e^{-k}}$$

$$= \frac{1}{1+e^{-kx}+e^{-k}e^{kx}+e^{-k}} = \frac{e^{kx}}{e^{kx}\left(1+e^{-k}\right)+1+e^{-k}e^{2kx}}$$

La fonction $x \longrightarrow e^{kx}\left(1+e^{-k}\right)+1+e^{-k}e^{2kx}$ est un polynôme en e^{kx} dont le discriminant est positif. Par conséquent la fraction rationnelle $g(x)$ admet une décomposition en éléments simples du premier ordre et il existe quatre réels η_1, η_2, δ_1, δ_2 tels que :

$$g(x) = \frac{\eta_1}{1+e^{kx+\delta_1}} + \frac{\eta_2}{1+e^{kx+\delta_2}}$$

Par conséquent :

$$f(y_1,\ldots,y_p) = \prod_{i=1}^{p} g(y_i) = \prod_{i=1}^{p}\left[\frac{\eta_1^i}{1+e^{ky_i+\delta_1^i}} + \frac{\eta_2^i}{1+e^{ky_i+\delta_2^i}}\right]$$

Il existe $n \in \mathbb{N}$ tel qu'il soit possible d'écrire f sous la forme :

$$f(y) = \sum_{i=1}^{n} \frac{\gamma_i}{1+e^{<x_i,y>+b_i}}$$

(C.2.2) □

Lemme C.2.3 : approximation d'une fonction indicatrice
Soit $C \subset \mathbb{R}^p, C$ compact, alors :

$$\forall \varepsilon > 0,\ \forall \alpha > 0,\ \exists\,(x_1,...,x_n) \in (\mathbb{R}^p)^n,\ \exists\,(b_1,...,b_n) \in \mathbb{R}^n \text{ tels que } \forall x \in \mathbb{R}^p,$$

$$\left|\sum_{i=1}^{n} \frac{\gamma_i}{1+e^{\langle x_i,x\rangle+b_i}} - \mathbb{1}_{\{x\in C\}}\right| \leqslant 1+2\varepsilon^2$$

$$\text{et} \quad \inf_{y\in Fr(C)} \|x-y\| > \alpha \Rightarrow \left|\sum_{i=1}^{n} \frac{\gamma_i}{1+e^{\langle x_i,x\rangle+b_i}} - \mathbb{1}_{\{x\in C\}}\right| \leqslant \varepsilon$$

Démonstration (lemme C.2.3) :

Partie A (démonstration de C.2.3)

Soit $C_1 = \{y = (y_1,...,y_p) \in \mathbb{R}^p \mid \forall i \in \{1,...,n\},\ 0 \leqslant y_i \leqslant 1\}$ et
$C_2^j = \{y = (y_1,...,y_p) \in \mathbb{R}^p \mid \forall i \neq j,\ 0 \leqslant y_i \leqslant 1 \text{ et } 1 \leqslant y_j \leqslant 2\}$

Le lemme C.2.2 suggère que la fonction cherchée pour ce lemme dans le cas particulier $C_1 \cup C_2^j$ est :

$$
\begin{aligned}
f(y_1,...,y_p) = & \prod_{i=1}^{p}\frac{1}{1+e^{-ky_i}}\prod_{i=1}^{p}\frac{1}{1+e^{-k(1-y_i)}} + \\
& \left(\prod_{i\neq j}\frac{1}{1+e^{-ky_i}}\right)\left(\prod_{i\neq j}\frac{1}{1+e^{-k(1-y_i)}}\right)\frac{1}{1+e^{k(1-y_j)}}\frac{1}{1+e^{-k(2-y_j)}} \\
= & \left(\prod_{i\neq j}\frac{1}{1+e^{-ky_i}}\right)\left(\prod_{i\neq j}\frac{1}{1+e^{-k(1-y_i)}}\right) \\
& \left(\frac{1}{1+e^{-ky_j}}\frac{1}{1+e^{-k(1-y_j)}}+\frac{1}{1+e^{k(1-y_j)}}\frac{1}{1+e^{-k(2-y_j)}}\right) \\
= & \left(\prod_{i\neq j}\frac{1}{1+e^{-ky_i}}\right)\left(\prod_{i\neq j}\frac{1}{1+e^{-k(1-y_i)}}\right) \\
& \left[\frac{1}{1+e^{-ky_j}}\left(\frac{1}{1+e^{-k(1-y_j)}}+1-1\right)+\left(1-\frac{1}{1+e^{-k(1-y_j)}}\right)\frac{1}{1+e^{-k(2-y_j)}}\right]
\end{aligned}
$$

Pour $k \geqslant k_0(\epsilon, \alpha, 2p)$, on a :

$$
\begin{aligned}
f(y_1,...,y_p) = & \left(\prod_{i\neq j}\frac{1}{1+e^{-ky_i}}\right)\left(\prod_{i\neq j}\frac{1}{1+e^{-k(1-y_i)}}\right) \\
& \left(\frac{1}{1+e^{-ky_j}}+\frac{1}{1+e^{-k(2-y_j)}}+\underbrace{\frac{1}{1+e^{k(1-y_j)}}\frac{1}{1+e^{-ky_j}}}_{\leqslant\varepsilon^2}-\underbrace{\frac{1}{1+e^{-k(1-y_j)}}\frac{1}{1+e^{-k(2-y_j)}}}_{\leqslant\varepsilon^2}\right)
\end{aligned}
$$

Par conséquent, il est facile de construire la fonction cherchée pour tout compact connexe par arc.

Partie B (démonstration de C.2.3)

Si un compact C n'est pas connexe par arc, on peut le recouvrir par une somme finie de compacts connexes par arcs et disjoints $(C_k)_{1\leqslant k\leqslant K}$ de telle sorte que :

$$\forall y \in \bigcup_{k=1}^{K} C_k,\ \inf\{\|x-y\|,\ x\in C\} \leqslant \frac{\alpha}{2}$$

(C.2.3) □

Démonstration (théorème C.2.1) :

Partie A (démonstration de C.2.1)

On démontre le théorème dans le cas où $q = 1$

Soit f une fonction continue du compact $C \subset \mathbb{R}^p \longrightarrow \mathbb{R}$, et soit $\varepsilon > 0$

On suppose également que f est positive, dans le cas contraire, on pose $f = \underbrace{f - \inf f}_{\text{fonction positive}} + \inf f$

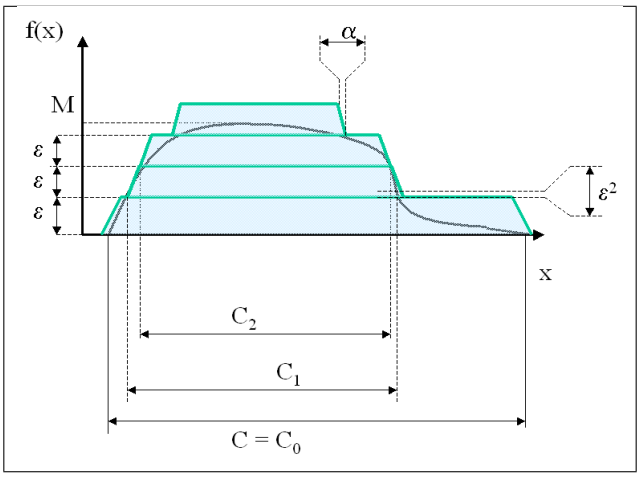


Fig. C.9: Idée de la démonstration du théorème de densité C.2.1.

Si f est nulle, alors c'est fini, sinon, on pose $M = \sup_{x \in C} f(x)$. M existe car f est continue et C est compact (de même, inf f existe également).

On pose $C_k = f^{-1}([k\varepsilon, M])$, C_k est compact car il est l'image réciproque d'un compact par une fonction continue et $C_k \subset C$ compact (voir figure C.9).

Par construction, $C_{k+1} \subset C_k$ et $C = \bigcup_{k=0}^{\frac{M}{\varepsilon}} C_k = C_0$, on définit :

$$\forall x \in C,\ g_\varepsilon(x) = \varepsilon \sum_{k=0}^{\frac{M}{\varepsilon}} \mathbb{1}_{\{x \in C_k\}}$$

D'où :

$$\begin{aligned} f(x) - g_\varepsilon(x) &= f(x) - \varepsilon \sum_{k=0}^{\frac{M}{\varepsilon}} \mathbb{1}_{\{x \in C_k\}} = f(x) - \varepsilon \sum_{k=0}^{\frac{M}{\varepsilon}} \mathbb{1}_{\{f(x) \geqslant k\varepsilon\}} \\ &= f(x) - \varepsilon \left[\frac{f(x)}{\varepsilon} \right] \quad \text{(partie entière)} \\ \text{d'où } \ 0 \leqslant f(x) - g_\varepsilon(x) &\leqslant \frac{\varepsilon}{4} \end{aligned} \tag{C.4}$$

Comme f est continue sur un compact, elle est uniformément continue sur ce compact :

$$\exists \alpha > 0 \text{ tel que } \forall (x,y) \in C^2,\ \|x - y\| \leqslant \alpha \Longrightarrow |f(x) - f(y)| \leqslant \tfrac{\varepsilon}{2}$$

$$\text{d'où } \ |f(x) - f(y)| \geqslant \varepsilon \Longrightarrow \|x - y\| > \alpha$$

Par conséquent :

$$\inf \{ \|x - y\| \mid x \in Fr(C_k),\ y \in Fr(C_{k+1}) \} > \alpha$$

D'après le lemme C.2.3, on peut construire des fonctions $h_k(x) = \sum_{i=1}^{n} \frac{1}{1 + e^{\langle x_i^k, x \rangle + b_i^k}}$ telles que :

$$\left(\left\|h_k(x) - \mathbb{1}_{\{x\in C_k\}}\right\| \leqslant 1\right) \text{ et } \left(\inf_{y\in Fr(C)} \|x-y\| > \frac{\alpha}{2} \Rightarrow \left\|h_k(x) - \mathbb{1}_{\{x\in C_k\}}\right\| \leqslant \varepsilon^2\right)$$

On en déduit que :

$$\begin{aligned}\left|f(x) - \varepsilon\sum_{k=0}^{\frac{M}{\varepsilon}} h_k(x)\right| &\leqslant |f(x) - g_\varepsilon(x)| + \left|g_\varepsilon(x) - \varepsilon\sum_{k=0}^{\frac{M}{\varepsilon}} h_k(x)\right| \\ &\leqslant \varepsilon + \varepsilon^2\left[\frac{M}{\varepsilon}\right] + 2\varepsilon^2 \\ &\leqslant \varepsilon(M+3)\end{aligned}$$

Comme $\varepsilon\sum_{k=1}^{\frac{M}{\varepsilon}} h_k(x)$ est de la forme désirée, le théorème est démontré dans le cas $q = 1$

Partie B (démonstration de C.2.1)

Dans le cas $q > 1$, on utilise la méthode précédente pour chacune des projections de f dans un repère orthonormé de $\mathbb{R}^q$. Il suffit de sommer sur chacune des dimensions.

(C.2.1) □

Ce théorème montre qu'il est judicieux de modéliser la fonction f dans l'équation C.3 par un réseau de neurones puisqu'il possible de s'approcher d'aussi près qu'on veut de la fonction $E(Y \mid X)$, il suffit d'ajouter des neurones sur la couche cachée du réseau. Ce théorème permet de déduire le corollaire suivant :

Corollaire C.2.4 : base
Soit F_p l'ensemble des fonctions continues de $C \subset \mathbb{R}^p \longrightarrow \mathbb{R}$ avec C compact muni de la norme :

$$\|f\| = \sup_{x\in C} \|f(x)\|$$

Alors l'ensemble E_p des fonctions sigmoïdes :

$$E_p = \left\{x \longrightarrow 1 - \frac{2}{1+e^{<y,x>+b}} \mid y \in \mathbb{R}^p \text{ et } b \in \mathbb{R}\right\}$$

est une base de F_p.

Démonstration (corollaire C.2.4) :

Le théorème C.2.1 montre que la famille E_p est une famille génératrice. Il reste à montrer que c'est une famille libre. Soient $(y_i)_{1\leqslant i\leqslant N} \in (\mathbb{R}^p)^N$ et $(b_i)_{1\leqslant i\leqslant N} \in \mathbb{R}^N$ vérifiant :

$$i \neq j \Longrightarrow y_i \neq y_j \text{ ou } b_i \neq b_j$$

Soit $(\lambda_i)_{1\leqslant i\leqslant N} \in \mathbb{R}^N$, il faut montrer que :

$$\forall x \in \mathbb{R}^p,\ \sum_{i=1}^{N} \lambda_i \left(1 - \frac{2}{1+e^{<y_i,x>+b_i}}\right) = 0 \Longrightarrow \forall i\ \lambda_i = 0 \tag{C.5}$$

(C.5) est évidemment pour $N = 1$. La démonstration est basée sur un raisonnement par récurrence, l'assertion (C.5) est supposée vraie pour $N-1$, démontrons qu'elle est vraie pour N. On suppose donc $N \geqslant 2$. S'il existe $i \in \{1,\dots,N\}$ tel que $y_i = 0$, la fonction $x \longrightarrow 1 - \dfrac{2}{1+e^{<y_i,x>+b_i}}$ est une constante, par conséquent, dans ce cas, (C.5) est vraie pour N. Dans le cas contraire, $\forall i \in \{1,\dots,N\}\,,\ y_i \neq 0$. On définit les vecteurs $X_i = (x_i, 1)$ et $Y_i = (y_j, b_j)$. On cherche à résoude le système de N équations à N inconnues :

$$\left\{\begin{array}{rcl} \sum\limits_{j=1}^{N} \lambda_j \left(1 - \dfrac{2}{1+e^{<Y_j,X_1>}}\right) &=& 0 \\ \dots & & \\ \sum\limits_{j=1}^{N} \lambda_j \left(1 - \dfrac{2}{1+e^{<Y_j,X_i>}}\right) &=& 0 \\ \dots & & \\ \sum\limits_{j=1}^{N} \lambda_j \left(1 - \dfrac{2}{1+e^{<Y_j,X_N>}}\right) &=& 0 \end{array}\right. \tag{C.6}$$

On note le vecteur $\Lambda = (\lambda_i)_{1\leqslant i\leqslant N}$ et M la matrice :

$$M = (m_{ij})_{1\leqslant i,j\leqslant N} = \left(1 - \frac{2}{1+e^{<Y_j,X_i>}}\right)_{1\leqslant i,j\leqslant N}$$

L'équation (C.6) est équivalente à l'équation matricielle : $M\Lambda = 0$. On effectue une itération du pivot de Gauss :

$$(C.6) \Longleftrightarrow \left\{\begin{array}{ccccccccc} \lambda_1 m_{11} & + & \lambda_2 & m_{12} & + & \dots & + & \lambda_N & m_{1N} & = 0 \\ 0 & + & \lambda_2 & (m_{22}m_{11} - m_{12}m_{21}) & + & \dots & + & \lambda_N & (m_{2N}m_{11} - m_{1N}m_{21}) & = 0 \\ \dots & & & & & & & & & \\ 0 & + & \lambda_2 & (m_{N2}m_{11} - m_{12}m_{N1}) & + & \dots & + & \lambda_N & (m_{NN}m_{11} - m_{1N}m_{N1}) & = 0 \end{array}\right.$$

On note $\Lambda_* = (\lambda_i)_{2\leqslant i\leqslant N}$ et Δ_*, M_* les matrices :

$$\begin{array}{rcl} M_* &=& (m_{ij})_{2\leqslant i,j\leqslant N} \\ \Delta_* &=& (m_{1j}\, m_{i1})_{2\leqslant i,j\leqslant N} \end{array}$$

Donc :

$$(C.6) \iff \left\{\begin{array}{ccccc} \lambda_1 m_{11} & + & \lambda_2 m_{12} + \dots + \lambda_N m_{1N} & = 0 \\ 0 & + & (m_{11} M_* - \Delta_*)\,\Lambda_* & = 0 \end{array}\right. \tag{C.7}$$

Il est possible de choisir $X_1(\alpha) = (\alpha x_1, 1)$ de telle sorte qu'il existe une suite $(s_l)_{1\leqslant l\leqslant N} \in \{-1,1\}^N$ avec $s_1 = 1$ et vérifiant :

$$\forall j \in (1,\dots,N)\,,\ \lim_{\alpha\longrightarrow+\infty} \left[1 - \frac{2}{1+e^{<Y_j,\,X_1(\alpha)>}}\right] = \lim_{\alpha\longrightarrow+\infty} m_{1j}(\alpha) = s_j$$

On définit :

$$\begin{array}{rcl} U_* & = & (m_{21}, \ldots, m_{N1})' \\ V_* & = & (s_2\, m_{21}, \ldots, s_N\, m_{N1})' \\ \text{et la matrice } L_* & = & (V_*)_{2 \leqslant i \leqslant N} \text{ dont les } N-1 \text{ colonnes sont identiques} \end{array}$$

On vérifie que :

$$\lim_{\alpha \longrightarrow +\infty} \Delta(\alpha) = V_*$$

On obtient :

$$\begin{aligned} (C.6) &\Longleftrightarrow \left\{ \begin{array}{ccccc} \lambda_1 m_{11}(\alpha) & + & \lambda_2 m_{12}(\alpha) + \ldots + \lambda_N m_{1N}(\alpha) & & = 0 \\ 0 & + & [m_{11}(\alpha)\, M_* - (L_* + (\Delta_*(\alpha) - L_*))]\, \Lambda_* & & = 0 \end{array} \right. \\ &\Longleftrightarrow \left\{ \begin{array}{ccccc} \lambda_1 m_{11}(\alpha) & + & \lambda_2 m_{12}(\alpha) + \ldots + \lambda_N m_{1N}(\alpha) & & = 0 \\ 0 & + & (m_{11}(\alpha)\, M_* - L_*)\, \Lambda_* + (\Delta_*(\alpha) - L_*)\, \Lambda_* & & = 0 \end{array} \right. \end{aligned} \tag{C.8}$$

On étudie la limite lorsque $\alpha \longrightarrow +\infty$:

$$\begin{array}{rcl} & (\Delta_*(\alpha) - L_*) & \underset{\alpha \to +\infty}{\longrightarrow} 0 \\ \Longrightarrow & (m_{11}(\alpha)\, M_* - L_*)\, \Lambda_* & \underset{\alpha \to +\infty}{\longrightarrow} 0 \\ \Longrightarrow & (M_* - L_*)\, \Lambda_* & = 0 \\ \Longrightarrow & M_* \Lambda_* - \left(\sum_{j=2}^{N} \lambda_j \right) V_* & = 0 \end{array}$$

Donc :

$$M_* \Lambda_* - \left(\sum_{j=2}^{N} \lambda_j \right) V_* = 0 \tag{C.9}$$

D'après l'hypothèse de récurrence, (C.9) implique que : $\forall i \in \{2, \ldots, N\},\ \lambda_i = 0$. Il reste à montrer que λ_1 est nécessairement nul ce qui est le cas losque $\alpha \longrightarrow +\infty$, alors $\lambda_1 m_{11}(\alpha) \longrightarrow \lambda_1 = 0$. La récurrence est démontrée.

(C.2.4) □

A chaque fonction sigmoïde du corollaire C.2.4 correspond un neurone de la couche cachée. Tous ont des rôles symétriques les uns par rapport aux autres ce qui ne serait pas le cas si les fonctions de transfert étaient des polynômes. C'est une des raisons pour lesquelles les réseaux de neurones ont du succès. Le théorème C.2.1 et le corollaire C.2.4 sont aussi vraies pour des fonctions du type exponentielle : $(y, b) \in \mathbb{R}^p \times \mathbb{R} \longrightarrow e^{-(<y,x>+b)^2}$. Maintenant qu'il est prouvé que les réseaux de neurones conviennent pour modéliser f dans l'équation (C.3), il reste à étudier les méthodes qui permettent de trouver les paramètres W^* optimaux de cette fonction.

C.3 Méthode d'optimisation de Newton

Lorsqu'un problème d'optimisation n'est pas soluble de manière déterministe, il existe des algorithmes permettant de trouver une solution approchée à condition toutefois que la fonction à maximiser ou minimiser

soit dérivable, ce qui est le cas des réseaux de neurones. Plusieurs variantes seront proposées regroupées sous le terme de descente de gradient.

C.3.1 Algorithme et convergence

Soit $g : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction dérivable dont il faut trouver $\overset{*}{x} = \underset{x \in \mathbb{R}}{\arg\min}\, g(x)$, le schéma C.10 illustre la méthode de descente de gradient dans le cas où $g(x) = x^2$.

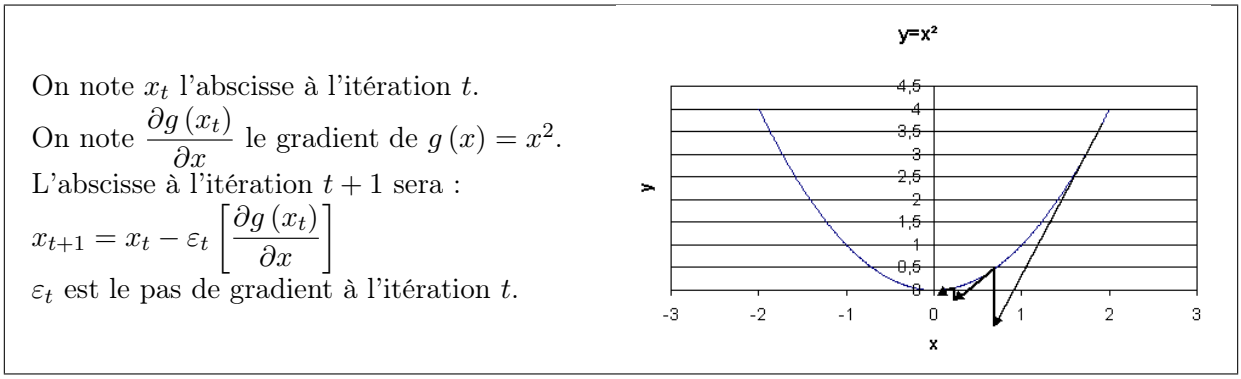


Fig. C.10: Minimisation par descente de gradient.

On suppose maintenant que g est une fonction dérivable $g : \mathbb{R}^q \longrightarrow \mathbb{R}$ dont il faut trouver le minimum, le théorème suivant démontre la convergence de l'algorithme de descente de gradient à condition que certaines hypothèses soient vérifiées. Une généralisation de ce théorème est présentée dans [Driancourt1996].

Théorème C.3.1 : convergence de la méthode de Newton (Bottou1991)
Soit une fonction continue $g : W \in \mathbb{R}^M \longrightarrow \mathbb{R}$, de classe C^1
On suppose les hypothèses suivantes vérifiées :

H1 $\underset{W \in \mathbb{R}^q}{\arg\min}\, g(W) = \{W^*\}$ est un singleton

H2 $\forall \varepsilon > 0, \inf_{|W - W^*| > \varepsilon} \left[(W - W^*)'.\nabla g(W)\right] > 0$

H3 $\exists (A, B) \in \mathbb{R}^2$ tels que $\forall W \in \mathbb{R}^p,\ \|\nabla g(W)\|^2 \leqslant A^2 + B^2 \|W - W^*\|^2$

H4 la suite $(\varepsilon_t)_{t \geqslant 0}$ vérifie, $\forall t > 0, \quad \varepsilon_t \in \mathbb{R}_+^*$ et $\sum_{t \geqslant 0} \varepsilon_t = +\infty, \quad \sum_{t \geqslant 0} \varepsilon_t^2 < +\infty$

Alors la suite $(W_t)_{t \geqslant 0}$ construite de la manière suivante :

$$W_0 \in \mathbb{R}^M \text{ et } \forall t \geqslant 0,\ W_{t+1} = W_t - \varepsilon_t \nabla g(W_t)$$

vérifie $\lim_{t \longrightarrow +\infty} W_t = W^*$

L'hypothèse *H1* implique que le minimum de la fonction g est unique et l'hypothèse *H2* implique que le demi-espace défini par l'opposé du gradient contienne toujours le minimum de la fonction g. L'hypothèse *H3* est vérifiée pour une fonction sigmoïde, elle l'est donc aussi pour toute somme finie de fonctions sigmoïdes que sont les réseaux de neurones à une couche cachée.

Démonstration (théorème C.3.1) :

Partie A (démonstration de C.3.1)

Soit la suite $u_t = \ln\left(1+\varepsilon_t^2 x^2\right)$ avec $x \in \mathbb{R}$, comme $\sum_{t\geqslant 0}\varepsilon_t^2 < +\infty,\ u_t \sim \varepsilon_t^2 x^2$, on a $\sum_{t\geqslant 0} u_t < +\infty$

Par conséquent, si $v_t = e^{u_t}$ alors $\prod_{t=1}^{T} v_t \overset{T\to\infty}{\longrightarrow} D \in \mathbb{R}$

Partie B (démonstration de C.3.1)

On pose $h_t = \|W_t - W^*\|^2$

Donc :

$$h_{t+1} - h_t = \|W_t - \varepsilon_t \nabla g\left(W_t\right) - W^*\|^2 - \|W_t - W^*\|^2 \tag{C.10}$$

Par conséquent :

$$h_{t+1} - h_t = -2\varepsilon_t \underbrace{(W_t - W^*)'\ \nabla g\left(W_t\right)}_{>0} + \varepsilon_t^2\ \|\nabla C\left(W_t\right)\|^2 \leqslant \varepsilon_t^2\ \|\nabla g\left(W_t\right)\|^2 \leqslant \varepsilon_t^2\ \left(A^2 + B^2 h_t\right)$$

D'où :

$$h_{t+1} - h_t\left(1+\varepsilon_t^2 B^2\right) \leqslant \varepsilon_t^2\, A^2$$

On pose $\pi_t = \prod_{k=1}^{t}\left(1+\varepsilon_k^2 B^2\right)^{-1}$ alors, en multipliant des deux côtés par π_{t+1}, on obtient :

$$\begin{aligned} \pi_{t+1}h_{t+1} - \pi_t h_t &\leqslant \varepsilon_t^2\, A^2 \pi_{t+1} \\ \text{d'où } \pi_{q+1}h_{q+1} - \pi_p h_p &\leqslant \sum_{t=p}^{q}\varepsilon_t^2\, A^2\pi_{t+1} \leqslant \sum_{t=p}^{q}\varepsilon_t^2\, A^2\Pi \leqslant \sum_{t=p}^{q}\varepsilon_t^2\, A^2\Pi \underset{t\longrightarrow\infty}{\longrightarrow} 0 \end{aligned}$$

Comme la série $\sum_t\left(\pi_{t+1}h_{t+1} - \pi_t h_t\right)$ vérifie le critère de Cauchy, elle est convergente. Par conséquent :

$$\lim_{q\to\infty}\pi_{q+1}h_{q+1} = 0 = \lim_{q\to\infty}\Pi h_{q+1}$$

D'où :

$$\lim_{q\to\infty} h_q = 0 \tag{C.11}$$

Partie C (démonstration de C.3.1)

La série $\sum_t\left(h_{t+1} - h_t\right)$ est convergente car $\Pi h_t \sim \pi_t h_t$.

$\sum_{t\geqslant 0}\varepsilon_t^2\ \|\nabla g\left(W_t\right)\|^2$ l'est aussi (d'après H3).

D'après (C.10), la série $\sum_{t\geqslant 0}\varepsilon_t \left(W_t - W^*\right)' \nabla g\left(W_t\right)$ est donc convergente. Or d'après les hypothèses (H2, H4), elle ne peut l'être que si :

$$\lim_{t\to\infty} W_t = W^* \tag{C.12}$$

(C.3.1) □

Si ce théorème prouve la convergence de la méthode de Newton, il ne précise pas à quelle vitesse cette convergence s'effectue et celle-ci peut parfois être très lente. Plusieurs variantes ont été développées regroupées sous le terme de méthodes de quasi-Newton dans le but d'améliorer la vitesse de convergence (voir paragraphe C.4).

Ce théorème peut être étendu dans le cas où la fonction g n'a plus un seul minimum global mais plusieurs minima locaux [Bottou1991], dans ce cas, la suite (W_t) converge vers un mimimum local. Dans le cas des réseaux de neurones, la fonction à optimiser est :

$$G\left(W\right) = \sum_{i=1}^{N} e\left(Y_i, \widehat{Y_i^W}\right) = \sum_{i=1}^{N} e\left(Y_i, f\left(W, X_i\right)\right) \tag{C.13}$$

Dès que les fonctions de transfert ne sont pas linéaires, il existe une multitude de minima locaux, ce nombre croissant avec celui des coefficients.

C.3.2 Calcul du gradient ou *rétropropagation*

Afin de minimiser la fonction G décrite en (C.13), l'algorithme de descente du gradient nécessite de calculer le gradient de cette fonction G qui est la somme des gradients $\frac{\partial e}{\partial W}$ pour chaque couple (X_i, Y_i) :

$$\begin{aligned}\frac{\partial G}{\partial W}\left(W\right) &= \sum_{i=1}^{N} \frac{\partial e\left(Y_i, f\left(W, X_i\right)\right)}{\partial W} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{C_C} \frac{\partial e\left(Y_i, f\left(W, X_i\right)\right)}{\partial z_{C,k}} \frac{\partial z_{C,k}}{\partial W}\end{aligned} \tag{C.14}$$

Les notations utilisées sont celles de la figure C.2 page 192. Les résultats qui suivent sont pour $X_i = X$ donné appartenant à la suite (X_i). On remarque tout d'abord que :

$$\frac{\partial e}{\partial w_{c,i,j}}\left(W, X\right) = z_{c-1,j} \frac{\partial e}{\partial y_{c,i}}\left(W, X\right) \tag{C.15}$$

$$\frac{\partial e}{\partial b_{c,i}}\left(W, X\right) = \frac{\partial e}{\partial y_{c,i}}\left(W, X\right) \tag{C.16}$$

La rétropropagation du gradient consiste donc à calculer les termes : $\frac{\partial e}{\partial y_{.,.}}\left(W, X\right)$ puisque le gradient s'en déduit facilement. La dernière couche du réseau de neurones nous permet d'obtenir :

$$\begin{aligned}\frac{\partial e}{\partial y_{C,i}}(W,X) &= \sum_{k=1}^{C_C} \frac{\partial e}{\partial z_{C,k}}(W,X)\,\frac{\partial z_{C,k}}{\partial y_{C,i}}(W,X) \\ &= \frac{\partial e}{\partial z_{C,i}}(W,X)\,f'_{c,i}(y_{C,i}) \end{aligned} \tag{C.17}$$

Pour les autres couches c telles que $1 \leqslant c \leqslant C-1$, on a :

$$\begin{aligned}\frac{\partial e}{\partial y_{c,i}} &= \sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}}\frac{\partial y_{c+1,l}}{\partial y_{c,i}} \\ &= \sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}}\left[\sum_{l=1}^{C_c} \frac{\partial y_{c+1,l}}{\partial z_{c,l}}\underbrace{\frac{\partial z_{c,l}}{\partial y_{c,i}}}_{=0\ si\ l\neq i}\right] \\ &= \sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}}\frac{\partial y_{c+1,l}}{\partial z_{c,i}}\frac{\partial z_{c,i}}{\partial y_{c,i}} \end{aligned} \tag{C.18}$$

Par conséquent :

$$\frac{\partial e}{\partial y_{c,i}} = \left[\sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}} w_{c+1,l,i}\right] f'_{c,i}(y_{c,i}) \tag{C.19}$$

Cette dernière formule permet d'obtenir par récurrence les dérivées $\frac{\partial e}{\partial y_{.,.}}$ de la dernière couche C à la première et ce, quel que soit le nombre de couches. Cette récurrence inverse de la propagation est appelée

rétropropagation. Cet algorithme se déduit des équations (C.14), (C.15), (C.16), (C.17) et (C.19) :

Algorithme C.3.2 : rétropropagation
Cet algorithme s'applique à un réseau de neurones vérifiant la définition C.1.3. Il s'agit de calculer sa dérivée par rapport aux poids. Il se déduit des formules (C.14), (C.15), (C.16), (C.17) et (C.19) et suppose que l'algorithme de propagation (C.1.4) a été préalablement exécuté.
On note $y'_{c,i} = \frac{\partial e}{\partial y_{c,i}}$, $w'_{c,i,j} = \frac{\partial e}{\partial w_{c,i,j}}$ et $b'_{c,i} = \frac{\partial e}{\partial b_{c,i}}$.

Etape A : initialisation
pour $i = 1$ **à** C_C **faire**
$y'_{C,i} \longleftarrow \frac{\partial e}{\partial z_{C,i}}(W, X)\, f'_{c,i}(y_{C,i})$
fin pour

Etape B : récurrence
pour $c = C - 1$ **à** 1 **faire**
pour $i = 1$ **à** C_c **faire**
$y'_{c,i} \longleftarrow 0$
pour $j = 1$ **à** C_{c+1} **faire**
$y'_{c,i} \longleftarrow y'_{c,i} + y'_{c+1,j}\, w_{c+1,j,i}$
fin pour
$y'_{c,i} \longleftarrow y'_{c,i}\, f'_{c,i}\left(y'_{c,i}\right)$
fin pour
fin pour

Etape C : terminaison
pour $c = 1$ **à** C **faire**
pour $i = 1$ **à** C_c **faire**
pour $j = 1$ **à** C_{c-1} **faire**
$w'_{c,i,j} \longleftarrow z_{c-1,j}\, y'_{c,i}$
$b'_{c,i,j} \longleftarrow y'_{c,i}$
fin pour
fin pour
fin pour

C.4 Apprentissage d'un réseau de neurones

Le terme apprentissage est encore inspiré de la biologie et se traduit par la minimisation de la fonction (C.13) où f est un réseau de neurone défini par (C.1.3). Il existe plusieurs méthodes pour effectuer celle-ci. Chacune d'elles vise à minimiser la fonction d'erreur :

$$E(W) = G(W) = \sum_{i=1}^{N} e\left(Y_i - \widehat{Y_i^W}\right) = \sum_{i=1}^{N} e\left(Y_i - f(W, X_i)\right)$$

Dans tous les cas, les différents apprentissages utilisent la suite suivante (ϵ_t) vérifiant (C.20) et proposent une convergence vers un minimum local (figure C.14).

$$\forall t > 0, \quad \varepsilon_t \in \mathbb{R}_+^* \text{ et } \sum_{t \geqslant 0} \varepsilon_t = +\infty, \quad \sum_{t \geqslant 0} \varepsilon_t^2 < +\infty \tag{C.20}$$

Il est souhaitable d'apprendre plusieurs fois la même fonction en modifiant les conditions initiales de ces méthodes de manière à améliorer la robustesse de la solution.

C.4.1 Apprentissage avec gradient global

L'algorithme C.3.2 permet d'obtenir la dérivée de l'erreur e pour un vecteur d'entrée X. Or l'erreur $E(W)$ à minimiser est la somme des erreurs pour chaque exemple X_i, le gradient global $\frac{\partial E(W)}{\partial W}$ de cette erreur globale est la somme des gradients pour chaque exemple (voir équation C.14). Parmi les méthodes d'optimisation basées sur le gradient global, on distingue deux catégories :

1. Les méthodes du premier ordre, elles sont calquées sur la méthode de Newton et n'utilisent que le gradient.
2. Les méthodes du second ordre ou méthodes utilisant un gradient conjugué, elles sont plus coûteuses en calcul mais plus performantes puisqu'elles utilisent la dérivée seconde ou une valeur approchée.

C.4.1.1 Méthodes du premier ordre

Les méthodes du premier ordre sont rarement utilisées. Elles sont toutes basées sur le principe de la descente de gradient de Newton présentée dans la section C.3 page 207 :

Algorithme C.4.1 : optimisation du premier ordre
Etape A : initialisation
Le premier jeu de coefficients W_0 du réseau de neurones est choisi aléatoirement.

$$\begin{aligned} t &\longleftarrow 0 \\ E_0 &\longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_0, X_i\right)\right) \end{aligned}$$

Etape B : calcul du gradient

$$g_t \longleftarrow \frac{\partial E_t}{\partial W}\left(W_t\right) = \sum_{i=1}^{N} e'\left(Y_i - f\left(W_t, X_i\right)\right)$$

Etape C : mise à jour

$$\begin{aligned} W_{t+1} &\longleftarrow W_t - \epsilon_t g_t \\ E_{t+1} &\longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right) \\ t &\longleftarrow t+1 \end{aligned}$$

Etape D : terminaison
si $\frac{E_t}{E_{t-1}} \approx 1$ (ou $\|g_t\| \approx 0$) alors l'apprentissage a convergé sinon retour à l'étape B

La condition d'arrêt peut-être plus ou moins stricte selon les besoins du problème. Cet algorithme converge vers un minimum local de la fonction d'erreur (d'après le théorème C.3.1) mais la vitesse de convergence est inconnue.

C.4.1.2 Méthodes du second ordre

L'algorithme C.4.1 fournit le canevas des méthodes d'optimisation du second ordre. La mise à jour des coefficients (étape C) est différente car elle prend en compte les dernières valeurs des coefficients ainsi que les derniers gradients calculés. Ce passé va être utilisé pour estimer une direction de recherche pour le minimum différente de celle du gradient, cette direction est appelée gradient conjugué (voir [Moré1977]).

Ces techniques sont basées sur une approximation du second degré de la fonction à minimiser. On note M le nombre de coefficients du réseau de neurones (biais compris). Soit $h : \mathbb{R}^M \longrightarrow \mathbb{R}$ la fonction d'erreur

associée au réseau de neurones :

$$h(W) = \sum_i e(Y_i, f(W, X_i))$$

Au voisinage de W_0, un développement limité donne :

$$h(W) = h(W_0) + \frac{\partial h(W_0)}{\partial W}(W - W_0) + (W - W_0)' \frac{\partial^2 h(W_0)}{\partial W^2}(W - W_0) + o\|W - W_0\|^2$$

Par conséquent, sur un voisinage de W_0, la fonction $h(W)$ admet un minimum local si $\frac{\partial^2 h(W_0)}{\partial W^2}$ est définie positive strictement [5]. Une matrice symétrique définie strictement positive est inversible, et le minimum est atteint pour la valeur :

$$W_{\min} = W_0 + \frac{1}{2}\left[\frac{\partial^2 h(W_0)}{\partial W^2}\right]^{-1}\left[\frac{\partial h(W_0)}{\partial W}\right] \tag{C.21}$$

Néanmoins, pour un réseau de neurones, le calcul de la dérivée seconde est coûteux, son inversion également. C'est pourquoi les dernières valeurs des coefficients et du gradient sont utilisées afin d'approcher cette dérivée seconde ou directement son inverse. Deux méthodes d'approximation sont présentées :

1. l'algorithme BFGS (Broyden-Fletcher-Goldfarb-Shano), [Broyden1967], [Fletcher1993]
2. l'algoritmhe DFP (Davidon-Fletcher-Powell), [Davidon1959], [Fletcher1963]

La figure C.11 est couramment employée pour illustrer l'intérêt des méthodes de gradient conjugué. Le problème consiste à trouver le minimum d'une fonction quadratique, par exemple, $G(x, y) = 3x^2 + y^2$. Tandis que le gradient est orthogonal aux lignes de niveaux de la fonction G, le gradient conjugué se dirige plus sûrement vers le minimum global.

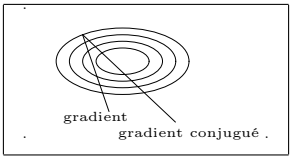


Fig. C.11: Gradient et gradient conjugué sur une ligne de niveau de la fonction $G(x, y) = 3x^2 + y^2$, le gradient est orthogonal aux lignes de niveaux de la fonction G, mais cette direction est rarement la bonne à moins que le point (x, y) se situe sur un des axes des ellipses, le gradient conjugué agrège les derniers déplacements et propose une direction de recherche plus plausible pour le minimum de la fonction.

Ces méthodes proposent une estimation de la dérivée seconde (ou de son inverse) utilisée en (C.21). Dans les méthodes du premier ordre, une itération permet de calculer les poids W_{t+1} à partir des poids W_t et du gradient G_t. Si ce gradient est petit, on peut supposer que G_{t+1} est presque égal au produit de la dérivée seconde par G_t. Cette relation est mise à profit pour construire une estimation de la dérivée seconde. Cette matrice notée B_t dans l'algorithme C.4.2 est d'abord supposée égale à l'identité puis actualisée à chaque

5. **Rappel :** $\frac{\partial^2 h(W_0)}{\partial W^2}$ est définie positive strictement $\Longleftrightarrow \forall Z \in \mathbb{R}^N,\ Z \neq 0 \Longrightarrow Z' \frac{\partial^2 h(W_0)}{\partial W^2} Z > 0$

itération en tenant de l'information apportée par chaque déplacement.

Algorithme C.4.2 : algorithme BFGS
Le nombre de paramètres de la fonction f est M.

Etape A : initialisation

Le premier jeu de coefficients W_0 du réseau de neurones est choisi aléatoirement.

$$\begin{array}{lcl} t & \longleftarrow & 0 \\ E_0 & \longleftarrow & \sum_{i=1}^{N} e\left(Y_i - f\left(W_0, X_i\right)\right) \\ B_0 & \longleftarrow & I_M \\ i & \longleftarrow & 0 \end{array}$$

Etape B : calcul du gradient

$$\begin{array}{lcl} g_t & \longleftarrow & \frac{\partial E_t}{\partial W}\left(W_t\right) = \sum_{i=1}^{N} e'\left(Y_i - f\left(W_t, X_i\right)\right) \\ c_t & \longleftarrow & B_t g_t \end{array}$$

Etape C : mise à jour des coefficients

$$\begin{array}{lcl} \epsilon^* & \longleftarrow & \arg\inf_{\epsilon} \sum_{i=1}^{N} e\left(Y_i - f\left(W_t - \epsilon c_t, X_i\right)\right) \\ W_{t+1} & \longleftarrow & W_t - \epsilon^* c_t \\ E_{t+1} & \longleftarrow & \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right) \\ t & \longleftarrow & t+1 \end{array}$$

Etape D : mise à jour de la matrice B_t

si $t - i \geqslant M$ **ou** $g'_{t-1} B_{t-1} g_{t-1} \leqslant 0$ **ou** $g'_{t-1} B_{t-1}\left(g_t - g_{t-1}\right) \leqslant 0$ **alors**

$$\begin{array}{lcl} B_t & \longleftarrow & I_M \\ i & \longleftarrow & t \end{array}$$

sinon

$$\begin{array}{lcl} s_t & \longleftarrow & W_t - W_{t-1} \\ d_t & \longleftarrow & g_t - g_{t-1} \\ B_t & \longleftarrow & B_{t-1} + \left(1 + \frac{d'_t B_{t-1} d_t}{d'_t s_t}\right) \frac{s_t s'_t}{s'_t d_t} - \frac{s_t d'_t B_{t-1} + B_{t-1} d_t s'_t}{d'_t s_t} \end{array}$$

fin si

Etape E : terminaison

si $\frac{E_t}{E_{t-1}} \approx 1$ alors l'apprentissage a convergé sinon retour à l'étape B

Lorsque la matrice B_t est égale à l'identité, le gradient conjugué est égal au gradient. Au fur et à mesure des itérations, cette matrice toujours symétrique évolue en améliorant la convergence de l'optimisation. Néanmoins, la matrice B_t doit être "nettoyée" (égale à l'identité) fréquemment afin d'éviter qu'elle n'agrège un passé trop lointain. Elle est aussi nettoyée lorsque le gradient conjugué semble trop s'éloigner du véritable gradient et devient plus proche d'une direction perpendiculaire.

La convergence de cet algorithme dans le cas des réseaux de neurones est plus rapide qu'un algorithme du premier ordre, une preuve en est donnée dans [Driancourt1996].

En pratique, la recherche de ϵ^* est réduite car le calcul de l'erreur est souvent coûteux, il peut être effectué

sur un grand nombre d'exemples. C'est pourquoi on remplace l'étape C par celle-ci les étapes B et D :

Algorithme C.4.3 : algorithme BFGS'
Le nombre de paramètre de la fonction f est M.
Etape A : initialisation
voir algorithme C.4.2

Etape B : calcul du gradient
voir algorithme C.4.2

Etape C : recherche de ϵ^*
$\epsilon^* \longleftarrow \epsilon_0$
faire
$\epsilon^* \longleftarrow \frac{\epsilon^*}{2}$
$W_{t+1} \longleftarrow W_t - \epsilon^* c_t$
$E_{t+1} \longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right)$
tant que $(E_{t+1} \geqslant E_t$ **et** $\epsilon^* \gg 0)$
si $\epsilon_* \approx 0$ **et** $B_t \neq I_M$ **alors**
$B_t \longleftarrow I_M$
$i \longleftarrow t$
retour à l'étape B
fin si

Etape D : mise à jour des coefficients
$W_{t+1} \longleftarrow W_t - \epsilon^* c_t$
$E_{t+1} \longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right)$
$t \longleftarrow t + 1$

Etape E : mise à jour de la matrice B_t
voir algorithme C.4.2

Etape F : terminaison
voir algorithme C.4.2

L'algorithme DFP est aussi un algorithme de gradient conjugué qui propose une approximation différente

de l'inverse de la dérivée seconde.

Algorithme C.4.4 : algorithme DFP
Le nombre de paramètre de la fonction f est M.
Etape A : initialisation
Le premier jeu de coefficients W_0 du réseau de neurones est choisi aléatoirement.

$$\begin{array}{lcl} t & \longleftarrow & 0 \\ E_0 & \longleftarrow & \sum_{i=1}^{N} e\left(Y_i - f\left(W_0, X_i\right)\right) \\ B_0 & \longleftarrow & I_M \\ i & \longleftarrow & 0 \end{array}$$

Etape B : calcul du gradient

$$\begin{array}{lcl} g_t & \longleftarrow & \frac{\partial E_t}{\partial W}\left(W_t\right) = \sum_{i=1}^{N} e'\left(Y_i - f\left(W_t, X_i\right)\right) \\ c_t & \longleftarrow & B_t g_t \end{array}$$

Etape C : mise à jour des coefficients

$$\begin{array}{lcl} \epsilon^* & \longleftarrow & \arg\inf_{\epsilon} \sum_{i=1}^{N} e\left(Y_i - f\left(W_t - \epsilon c_t, X_i\right)\right) \\ W_{t+1} & \longleftarrow & W_t - \epsilon^* c_t \\ E_{t+1} & \longleftarrow & \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right) \\ t & \longleftarrow & t+1 \end{array}$$

Etape D : mise à jour de la matrice B_t
si $t - i \geqslant M$ **ou** $g'_{t-1} B_{t-1} g_{t-1} \leqslant 0$ **ou** $g'_{t-1} B_{t-1}\left(g_t - g_{t-1}\right) \leqslant 0$ **alors**

$$\begin{array}{lcl} B_t & \longleftarrow & I_M \\ i & \longleftarrow & t \end{array}$$

sinon

$$\begin{array}{lcl} d_t & \longleftarrow & W_t - W_{t-1} \\ s_t & \longleftarrow & g_t - g_{t-1} \\ B_t & \longleftarrow & B_{t-1} + \frac{d_t d'_t}{d'_t s_t} - \frac{B_{t-1} s_t s'_t B_{t-1}}{s'_t B_{t-1} s_t} \end{array}$$

fin si
Etape E : terminaison
si $\frac{E_t}{E_{t-1}} \approx 1$ alors l'apprentissage a convergé sinon retour à l'étape B

Seule l'étape D de mise à jour de B_t diffère dans les algorithmes C.4.2 et C.4.4. Comme l'algorithme BFGS (C.4.2), on peut construire une version DFP' inspirée de l'algorithme C.4.3.

C.4.2 Apprentissage avec gradient stochastique

Compte tenu des courbes d'erreurs très "accidentées" (figure C.12) dessinées par les réseaux de neurones, il existe une multitude de minima locaux. De ce fait, l'apprentissage global converge rarement vers le minimum global de la fonction d'erreur lorsqu'on applique les algorithmes basés sur le gradient global. L'apprentissage avec gradient stochastique est une solution permettant de mieux explorer ces courbes d'erreurs. De plus, les méthodes de gradient conjugué nécessite le stockage d'une matrice trop grande parfois pour des fonctions ayant quelques milliers de paramètres. C'est pourquoi l'apprentissage avec gradient stochastique est souvent préféré à l'apprentissage global pour de grands réseaux de neurones

alors que les méthodes du second ordre trop coûteuses en calcul sont cantonnées à de petits réseaux. En contrepartie, la convergence est plus lente. La démonstration de cette convergence nécessite l'utilisation de quasi-martingales et est une convergence presque sûre [Bottou1991].

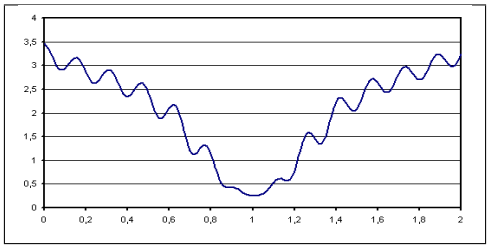


Fig. C.12: Exemple de minima locaux.

Algorithme C.4.5 : apprentissage stochastique
Etape A : initialisation
Le premier jeu de coefficients W_0 du réseau de neurones est choisi aléatoirement.

$$\begin{aligned} t &\longleftarrow 0 \\ E_0 &\longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_0, X_i\right)\right) \end{aligned}$$

Etape B : récurrence
$W_{t,0} \longleftarrow W_0$
pour $t' = 0$ **à** $N-1$ **faire**

$$\begin{aligned} i &\longleftarrow \text{ nombre aléatoire dans } \{1,\ldots,N\} \\ g &\longleftarrow \frac{\partial E}{\partial W}\left(W_{t,t'}\right) = e'\left(Y_i - f\left(W_{t,t'}, X_i\right)\right) \\ W_{t,t'+1} &\longleftarrow W_{t,t'} - \epsilon_t g \end{aligned}$$

fin pour

$$\begin{aligned} W_{t+1} &\longleftarrow W_{t,N} \\ E_{t+1} &\longleftarrow \sum_{i=1}^{N} e\left(Y_i - f\left(W_{t+1}, X_i\right)\right) \\ t &\longleftarrow t+1 \end{aligned}$$

Etape C : terminaison
si $\frac{E_t}{E_{t-1}} \approx 1$ alors l'apprentissage a convergé sinon retour à l'étape B.

En pratique, il est utile de converser le meilleur jeu de coefficients : $W^* = \arg\min_{u \geqslant 0} E_u$ car la suite $(E_u)_{u \geqslant 0}$ n'est pas une suite décroissante.

C.5 Classification

C.5.1 Vraisemblance d'un échantillon de variable suivant une loi multinomiale

Soit $(Y_i)_{1 \leqslant i \leqslant N}$ un échantillon de variables aléatoires i.i.d. suivant la loi multinomiale $\mathcal{M}(p_1, \ldots, p_C)$. On définit :

$$\forall k \in \{1, \cdots, C\},\ d_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{Y_i = k\}}$$

La vraisemblance de l'échantillon est :

$$\begin{aligned}
L(Y_1, \ldots, Y_N, p_1, \ldots, p_C) &= \prod_{i=1}^{n} p_{Y_i} \\
\ln L(Y_1, \ldots, Y_N, p_1, \ldots, p_C) &= \sum_{i=1}^{n} \ln p_{Y_i} \\
\ln L(Y_1, \ldots, Y_N, p_1, \ldots, p_C) &= \sum_{k=1}^{C} \left[(\ln p_k) \sum_{i=1}^{N} \mathbb{1}_{\{Y_i = k\}} \right] \\
\ln L(Y_1, \ldots, Y_N, p_1, \ldots, p_C) &= N \sum_{k=1}^{C} d_k \ln p_k
\end{aligned} \tag{C.22}$$

Cette fonction est aussi appelée distance de Kullback-Leiber ([Kullback1951]), elle mesure la distance entre deux distributions de variables aléatoires discrètes. L'estimateur de maximum de vraisemblance (emv) est la solution du problème suivant :

Problème C.5.1 : estimateur du maximum de vraisemblance
Soit un vecteur $(d_1, \ldots, d_N)$ tel que :

$$\begin{cases} \sum_{k=1}^{N} d_k = 1 \\ \forall k \in \{1, \ldots, N\},\ d_k \geqslant 0 \end{cases}$$

On cherche le vecteur $(p_1^*, \ldots, p_N^*)$ vérifiant :

$$(p_1^*, \ldots, p_N^*) = \underset{(p_1, \ldots, p_C) \in \mathbb{R}^C}{\arg\max} \sum_{k=1}^{C} d_k \ln p_k$$

$$\text{avec} \begin{cases} \forall k \in \{1, \cdots, C\},\ p_k \geqslant 0 \\ \text{et } \sum_{k=1}^{C} p_k = 1 \end{cases}$$

Théorème C.5.2 : résolution du problème C.5.1
La solution du problème C.5.1 est le vecteur :

$$(p_1^*, \ldots, p_N^*) = (d_1, \ldots, d_N)$$

Démonstration (théorème C.5.2) :

Soit un vecteur $(p_1, \ldots, p_N)$ vérifiant les conditions :

$$\left\{ \begin{array}{l} \sum\limits_{k=1}^{N} p_k = 1 \\ \forall k \in \{1, \ldots, N\}\,,\ p_k \geqslant 0 \end{array} \right.$$

La fonction $x \longrightarrow \ln x$ est concave, d'où :

$$\begin{aligned} \Delta &= \sum_{k=1}^{C} d_k \ln p_k - \sum_{k=1}^{C} d_k \ln d_k \\ &= \sum_{k=1}^{C} d_k \left(\ln p_k - \ln d_k\right) = \sum_{k=1}^{C} d_k \ln \frac{p_k}{d_k} \\ &\leqslant \ln \left(\sum_{k=1}^{C} d_k \frac{p_k}{d_k} \right) = \ln \left(\sum_{k=1}^{C} p_k \right) = \ln 1 = 0 \\ &\leqslant 0 \end{aligned}$$

(C.5.2) □

La distance de KullBack-Leiber compare deux distributions de probabilités entre elles. C'est elle qui va faire le lien entre le problème de classification discret (C.1.7) et les réseaux de neurones pour lesquels il faut impérativement une fonction d'erreur dérivable.

C.5.2 Problème de classification pour les réseaux de neurones

Le problème de classification C.1.7 est un cas particulier de celui qui suit pour lequel il n'est pas nécessaire de connaître la classe d'appartenance de chaque exemple mais seulement les probabilités d'appartenance de cet exemple à chacune des classes.

Soient une variable aléatoire continue $X \in \mathbb{R}^p$ et une variable aléatoire discrète multinomiale $Y \in \{1, \cdots, C\}$, on veut estimer la loi de :

$$Y|X \sim \mathcal{M}\left(p_1\left(W, X\right), \ldots, p_C\left(W, X\right)\right) \text{ avec } W \in \mathbb{R}^M$$

Le vecteur $\left(p_1\left(W, X\right), \ldots, p_C\left(W, X\right)\right)$ est une fonction f de (W, X) où W est l'ensemble des M paramètres du modèle. Cette fonction possède p entrées et C sorties. Comme pour le problème de la régression, on cherche les poids W qui correspondent le mieux à l'échantillon :

$$A = \left\{ \left(X_i, y_i = \left(\eta_i^k\right)_{1 \leqslant k \leqslant C} \right) \in \mathbb{R}^p \times [0,1]^C \text{ tel que } \sum_{k=1}^{c} y_i^k = 1 \middle| \, 1 \leqslant i \leqslant N \right\}$$

On suppose que les variables $\left(Y_i|X_i\right)_{1 \leqslant i \leqslant N}$ suivent les lois respectives $\left(\mathcal{M}\left(y_i\right)\right)_{1 \leqslant i \leqslant N}$ et sont indépendantes entre elles, la vraisemblance du modèle vérifie d'après l'équation (C.22) :

$$\begin{aligned} L_W &\propto \prod_{i=1}^{N} \prod_{k=1}^{C} \left[p_k\left(W, X_i\right)\right]^{\mathbb{P}(Y_i = k)} \\ \ln L_W &\propto \sum_{i=1}^{N} \sum_{k=1}^{C} \eta_i^k \ln \left[p_k\left(W, X_i\right)\right] \end{aligned}$$

La solution du problème $\overset{*}{W} = \underset{W \in \mathbb{R}^l}{\arg\max}\, L_W$ est celle d'un problème d'optimisation sous contrainte. Afin de contourner ce problème, on définit la fonction f :

$$\begin{aligned}
&f : \mathbb{R}^M \times \mathbb{R}^p \longrightarrow \mathbb{R}^C \\
&\forall\, (W, x) \in \mathbb{R}^M \times \mathbb{R}^p,\ f\,(W, x) = (f_1\,(W, x)), \ldots, f_C\,(W, x) \\
&\text{et } \forall i \in \{1, \cdots, N\},\ \forall k \in \{1, \cdots, C\},\ p^k\,(W, X_i) = \frac{e^{f_k(W, X_i)}}{\sum\limits_{l=1}^{C} e^{f_l(W, X_i)}}
\end{aligned}$$

Les contraintes sur $\left(p^k\,(W, X_i)\right)$ sont bien vérifiées :

$$\begin{aligned}
&\forall i \in \{1, \cdots, N\},\ \forall k \in \{1, \cdots, C\},\ p^k\,(W, X_i) \geqslant 0 \\
&\forall i \in \{1, \cdots, N\},\ \sum_{k=1}^{C} p^k\,(W, X_i) = 1
\end{aligned}$$

On en déduit que :

$$\begin{aligned}
\ln L_W &\propto \sum_{i=1}^{N} \sum_{k=1}^{C} \eta_i^k \left[f_k\,(W, X_i) - \ln \left[\sum_{l=1}^{C} e^{f_l(W, X_i)} \right] \right] \\
\ln L_W &\propto \sum_{i=1}^{N} \sum_{k=1}^{C} \eta_i^k f_k\,(W, X_i) - \sum_{i=1}^{N} \ln \left[\sum_{l=1}^{C} e^{f_l(W, X_i)} \right] \underbrace{\sum_{k=1}^{C} \eta_i^k}_{=1}
\end{aligned}$$

D'où :

$$\ln L_W \propto \sum_{i=1}^{N} \sum_{k=1}^{C} \eta_i^k f_k\,(W, X_i) - \sum_{i=1}^{N} \ln \left[\sum_{l=1}^{C} e^{f_l(W, X_i)} \right] \tag{C.23}$$

Ceci mène à la définition du problème de classification suivant :

Problème C.5.3 : classification
Soit A l'échantillon suivant :

$$A = \left\{ \left(X_i, y_i = \left(\eta_i^k \right)_{1 \leqslant k \leqslant C} \right) \in \mathbb{R}^p \times \mathbb{R}^C \text{ tel que } \sum_{k=1}^{c} \eta_i^k = 1 \middle| \, 1 \leqslant i \leqslant N \right\}$$

y_i^k représente la probabilité que l'élément X_i appartiennent à la classe k :

$$\eta_i^k = \mathbb{P}\left(Y_i = k \mid X_i \right)$$

Le classifieur cherché est une fonction f définie par :

$$\begin{array}{rcl} f : \mathbb{R}^M \times \mathbb{R}^p & \longrightarrow & \mathbb{R}^C \\ (W, X) & \longrightarrow & \left(f_1(W, X), \ldots, f_p(W, X) \right) \end{array}$$

Dont le vecteur de poids W^* est égal à :

$$W^* = \arg\max_W \sum_{i=1}^{N} \sum_{k=1}^{C} \eta_i^k f_k(W, X_i) - \sum_{i=1}^{N} \ln \left[\sum_{l=1}^{C} e^{f_l(W, X_i)} \right]$$

C.5.3 Réseau de neurones adéquat

Dans le problème précédent, la maximisation de $\overset{*}{W} = \arg\max_{W \in \mathbb{R}^M} L_W$ aboutit au choix d'une fonction :

$$X \in \mathbb{R}^p \longrightarrow f(\overset{*}{W}, X) \in \mathbb{R}^C$$

Le réseau de neurones (voir figure C.13) $g : (W, X) \in \mathbb{R}^M \times \mathbb{R}^p \longrightarrow \mathbb{R}^C$ choisi pour modéliser f aura pour sorties :

$$\begin{array}{l} X \in \mathbb{R}^p \longrightarrow g(\overset{*}{W}, X) \in \mathbb{R}^C \\ \forall k \in \{1, \cdots, C\}, \; g_k(W, X) = e^{f_k(W, X)} \end{array}$$

Les conséquences sont :

1. la fonction de transert des neurones de la couche de sortie est : $x \longrightarrow e^x$
2. la probabilité pour le vecteur $x \in \mathbb{R}^p$ d'appartenir à la classe $k \in \{1, \cdots, C\}$ est :

$$p_k(\overset{*}{W}, x) = \mathbb{P}\left(Y = k | x \right) = \frac{g_k(\overset{*}{W}, x)}{\sum_{l=1}^{C} g_l(\overset{*}{W}, x)}$$

3. la fonction d'erreur à minimiser est l'opposé de la log-vraisemblance du modèle :

$$\begin{aligned} \overset{*}{W} &= \arg\min_{W \in \mathbb{R}^M} \left[\sum_{i=1}^{N} \left(- \sum_{k=1}^{C} \eta_i^k \ln\left(g_k(W, X_i) \right) + \ln \left[\sum_{l=1}^{C} g_l(W, X_i) \right] \right) \right] \\ &= \arg\min_{W \in \mathbb{R}^M} \left[\sum_{i=1}^{N} h\left(W, X_i, \eta_i^k \right) \right] \end{aligned}$$

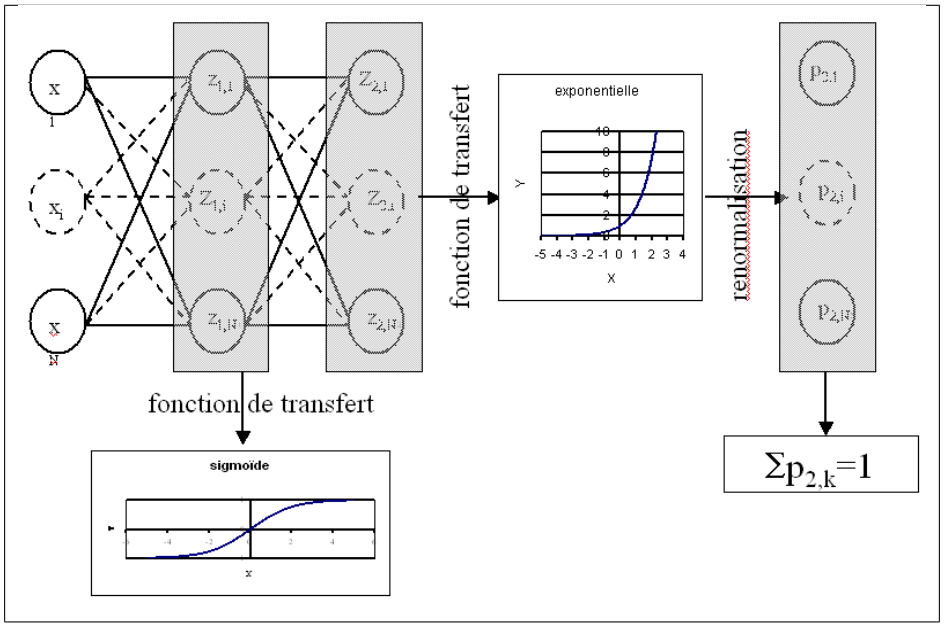


Fig. C.13: Réseau de neurones adéquat pour la classification.

On note C_{rn} le nombre de couches du réseau de neurones, $z^k_{C_{rn}}$ est la sortie k avec $k \in \{1, \cdots, C\}$, $g_k(W, x) = z^k_{C_{rn}} = e^{y^k_{C_{rn}}}$ où $y^k_{C_{rn}}$ est le potentiel du neurone k de la couche de sortie.
On calcule :

$$\begin{aligned}\frac{\partial h\left(W, X_i, y_i^k\right)}{\partial y^k_{C_{rn}}} &= -\eta_i^k + \frac{z^i_{Crn}}{\sum\limits_{m=1}^{C} z^m_{Crn}} \\ &= p_k(\overset{*}{W}, x) - \eta_i^k\end{aligned}$$

Cette équation permet d'adapter l'algorithme C.3.2 décrivant rétropropagation pour le problème de la

classification et pour un exemple $\left(X, y = \left(\eta^k\right)_{1 \leqslant k \leqslant C}\right)$. Seule la couche de sortie change.

Algorithme C.5.4 : rétropropagation
Cet algorithme de rétropropagation est l'adaptation de C.3.2 pour le problème de la classification.
Il suppose que l'algorithme de propagation (C.1.4) a été préalablement exécuté.
On note $y'_{c,i} = \frac{\partial e}{\partial y_{c,i}}$, $w'_{c,i,j} = \frac{\partial e}{\partial w_{c,i,j}}$ et $b'_{c,i} = \frac{\partial e}{\partial b_{c,i}}$.

Etape A : initialisation
pour $i = 1$ **à** C_C **faire**

$$y'_{C,i} \longleftarrow \frac{z_{C,i}}{\sum_{l=1}^{C} z_{C,l}} - \eta_i$$

fin pour

Etape B : récurrence
voir étape B et l'algorithme C.3.2

Etape C : terminaison
voir étape C et l'algorithme C.3.2

Remarque C.5.5: nullité du gradient
On vérifie que le gradient s'annule lorsque le réseau de neurones retourne pour l'exemple (X_i, y_i) la distribution de $Y|X_i \sim \mathcal{M}(y_i)$.

Remarque C.5.6: probabilité de sortie
L'algorithme de rétropropagation C.5.4 utilise un vecteur désiré de probabilités $(\eta_1, \ldots, \eta_{C_C})$ vérifiant $\sum_{i=1}^{C_C} \eta_i = 1$. L'expérience montre qu'il est préférable d'utiliser un vecteur vérifiant la contrainte :

$$\begin{aligned} &\forall i \in \{1, \ldots, C_C\}, \ \min\{\eta_i, 1 - \eta_i\} > \alpha \\ &\text{avec } \alpha > 0 \end{aligned} \tag{C.24}$$

Généralement, α est de l'ordre de $0,1$ ou $0,01$. Cette contrainte facilite le calcul de la vraisemblance (C.23) et évite l'obtention de gradients quasi-nuls qui freinent l'apprentissage lorsque les fonctions exponnetielles sont saturées (voir [Bishop1995]).

C.6 Prolongements

C.6.1 Base d'apprentissage et base de test

Les deux exemples de régression et de classification (paragraphes C.1.4 et C.1.5) ont montré que la structure du réseau de neurones la mieux adaptée a une grande importance. Dans ces deux cas, une rapide vérification visuelle permet de juger de la qualité du modèle obtenu après apprentissage, mais bien souvent, cette "vision" est inaccessible pour des dimensions supérieures à deux. Le meilleur moyen de jauger le modèle appris est de vérifier si l'erreur obtenue sur une base ayant servi à l'apprentissage (ou *base d'apprentissage*) est conservée sur une autre base (ou *base de test*) que le modèle découvre pour la première fois.

Soit $B = \{(X_i, Y_i) \mid 1 \leqslant i \leqslant N\}$ l'ensemble des observations disponibles. Cet ensemble est aléatoirement scindé en deux sous-ensembles B_a et B_t de telle sorte que :

$B_a \neq \emptyset$ et $B_t \neq \emptyset$
$B_a \cup B_t = B$ et $B_a \cap B_t = \emptyset$
$\frac{card(B_a)}{card(B_a \cup B_t)} = p \in]0,1[$, en règle générale, $p \in \left[\frac{1}{2}, \frac{3}{4}\right]$

Ce découpage est valide si tous les exemples de la base B obéissent à la même loi, les deux bases B_a et B_t sont dites *homogènes*. Le réseau de neurones sera donc appris sur la base d'apprentissage B_a et "testé" sur la base de test B_t. Le test consiste à vérifier que l'erreur sur B_t est sensiblement égale à celle sur B_a, auquel cas on dit que le modèle (ou réseau de neurones) généralise bien. Le modèle trouvé n'est pas pour autant le bon modèle mais il est robuste. La courbe figure C.14 illustre une définition du modèle optimal comme étant celui qui minimise l'erreur sur la base de test. Lorsque le modèle choisi n'est pas celui-là, deux cas sont possibles :

1. Le nombre de coefficients est trop petit : le modèle généralise bien mais il existe d'autres modèles meilleurs pour lesquels l'erreur d'apprentissage et de test est moindre.
2. Le nombre de coefficients est trop grand : le modèle généralise mal, l'erreur d'apprentissage est faible et l'erreur de test élevée, le réseau a appris la base d'apprentissage par cœur.

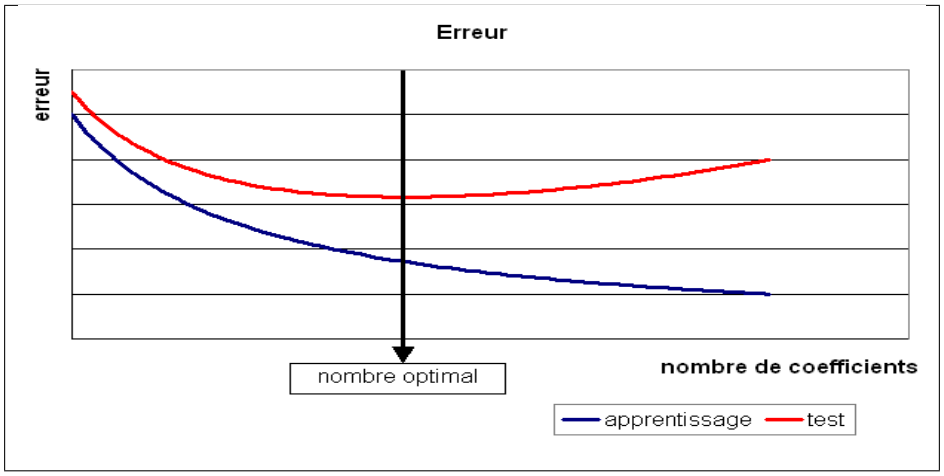


Fig. C.14: Modèle optimal pour la base de test

Ce découpage des données en deux bases d'apprentissage et de test est fréquemment utilisé pour toute estimation de modèles résultant d'une optimisation réalisée au moyen d'un algorithme itératif. C'est le cas par exemple des modèles de Markov cachés [6]. Elle permet de s'assurer qu'un modèle s'adapte bien à de nouvelles données.

C.6.2 Fonction de transfert à base radiale

La fonction de transfert est dans ce cas à base radiale (souvent abrégée par RBF pour "radial basis function"). Elle ne s'applique pas au produit scalaire entre le vecteur des poids et celui des entrées mais

6. Annexes : voir paragraphe E, page 245

à la distance euclidienne entre ces vecteurs.

Définition C.6.1 : neurone distance
Un neurone distance à p entrées est une fonction $f : \mathbb{R}^{p+1} \times \mathbb{R}^p \longrightarrow \mathbb{R}$ définie par :

1. $g : \mathbb{R} \longrightarrow \mathbb{R}$
2. $W \in \mathbb{R}^{p+1}$, $W = (w_1, \ldots, w_{p+1}) = (W', w_{p+1})$
3. $\forall x \in \mathbb{R}^p,\ f(W, x) = e^{-\|W'-x\|^2 + w_{p+1}}$
   avec $x = (x_1, \ldots, x_p)$

Ce neurone est un cas particulier du suivant qui pondère chaque dimension par un coefficient. Toutefois, ce neurone possède $2p+1$ coefficients où p est le nombre d'entrée.

Définition C.6.2 : neurone distance pondérée
Pour un vecteur donné $W \in \mathbb{R}^p = (w_1, \ldots, w_p)$, on note $W_i^j = (w_i, \ldots, w_j)$. Un neurone distance pondérée à p entrées est une fonction $f : \mathbb{R}^{2p+1} \times \mathbb{R}^p \longrightarrow \mathbb{R}$ définie par :

1. $g : \mathbb{R} \longrightarrow \mathbb{R}$
2. $W \in \mathbb{R}^{2p+1}$, $W = (w_1, \ldots, w_{2p+1}) = (w_1, w_{2p+1})$
3. $\forall x \in \mathbb{R}^p,\ f(W, x) = \exp\left[-\left[\sum_{i=1}^{p} w_{p+i}(w_i - x_i)^2\right] + w_{p+1}\right]$
   avec $x = (x_1, \ldots, x_p)$

La fonction de transfert est $x \longrightarrow e^x$ est le potentiel de ce neurone donc :

$$y = -\left[\sum_{i=1}^{p} w_{p+i}(w_i - x_i)^2\right] + w_{p+1}$$

L'algorithme de rétropropagation C.3.2 est modifié par l'insertion d'un tel neurone dans un réseau ainsi que la rétropropagation. Le plus simple tout d'abord :

$$1 \leqslant i \leqslant p,\ \frac{\partial y}{\partial w_i} = -2w_{p+i}(w_i - x_i) \tag{C.25}$$

$$p+1 \leqslant i \leqslant 2p,\ \frac{\partial y}{\partial w_i} = -(w_i - x_i)^2 \tag{C.26}$$

$$i = 2p+1,\ \frac{\partial y}{\partial w_i} = -1 \tag{C.27}$$

Pour le neurone distance simple, la ligne (C.26) est superflue, tous les coefficients $(w_i)_{p+1 \leqslant i \leqslant 2p}$ sont égaux à 1. La relation (C.18) reste vraie mais n'aboutit plus à (C.19), celle-ci devient en supposant que la couche d'indice $c+1$ ne contient que des neurones définie par (C.6.2) :

$$\begin{aligned}\frac{\partial e}{\partial y_{c,i}} &= \sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}} \frac{\partial y_{c+1,l}}{\partial z_{c,i}} \frac{\partial z_{c,i}}{\partial y_{c,i}} \\ &= \left[\sum_{l=1}^{C_{c+1}} \frac{\partial e}{\partial y_{c+1,l}} \left(2w_{c+1,l,p+i}(w_{c+1,l,i} - z_{c,i})\right)\right] \frac{\partial z_{c,i}}{\partial y_{c,i}}\end{aligned} \tag{C.28}$$

C.6.3 Poids partagés

Les poids partagés sont simplement un ensemble de poids qui sont contraints à conserver la même valeur. Soit G un groupe de poids partagés dont la valeur est w_G. Soit X_k et Y_k un exemple de la base d'apprentissage (entrées et sorties désirées), l'erreur commise par le réseau de neurones est $e\left(W, X_k, Y_k\right)$.

$$\frac{\partial e\left(W, X_k, Y_k\right)}{\partial w_G} = \sum_{w \in G} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial w_G} \frac{\partial w_G}{\partial w} = \sum_{w \in G} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial w_G}$$

Par conséquent, si un poids w appartient à un groupe G de poids partagés, sa valeur à l'itération suivante sera :

$$w_{t+1} = w_t - \varepsilon_t \left(\sum_{w \in G} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial w} \right)$$

C.6.4 Dérivée par rapport aux entrées

On note (X_k, Y_k) un exemple de la base d'apprentissage. Le réseau de neurones est composé de C couches, C_i est le nombre de neurones sur la i^{ime} couche, C_0 est le nombre d'entrées. Les entrées sont appelées $(z_{0,i})_{1 \leqslant i \leqslant C_0}$, $(y_{1,i})_{1 \leqslant i \leqslant C_1}$ sont les potentiels des neurones de la première couche, on en déduit que, dans le cas d'un neurone classique (non distance) :

$$\frac{\partial e\left(W, X_k, Y_k\right)}{\partial z_{0,i}} = \sum_{j=1}^{C_1} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial y_{1,j}} \frac{\partial y_{1,j}}{\partial z_{0,i}} = \sum_{j=1}^{C_1} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial y_{1,j}} w_{1,j,i}$$

Comme le potentiel d'un neurone distance n'est pas linéaire par rapport aux entrées $\left(y = \sum_{i=1}^{N} \left(w_i - z_{0,i}\right)^2 + b \right)$, la formule devient dans ce cas :

$$\frac{\partial e\left(W, X_k, Y_k\right)}{\partial z_{0,i}} = \sum_{j=1}^{C_1} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial y_{1,j}} \frac{\partial y_{1,j}}{\partial z_{0,i}} = -2 \sum_{j=1}^{C_1} \frac{\partial e\left(W, X_k, Y_k\right)}{\partial y_{1,j}} \left(w_{1,j,i} - z_{0,i}\right)$$

C.6.5 Régularisation ou Decay

Lors de l'apprentissage, comme les fonctions de seuil du réseau de neurones sont bornées, pour une grande variation des coefficients, la sortie varie peu. De plus, pour ces grandes valeurs, la dérivée est quasi nulle et l'apprentissage s'en trouve ralenti. Par conséquent, il est préférable d'éviter ce cas et c'est pourquoi un terme de régularisation est ajouté lors de la mise à jour des coefficients (voir [Bishop1995]). L'idée consiste à ajouter à l'erreur une pénalité fonction des coefficients du réseau de neurones :

$$E_{reg} = E + \lambda \sum_i w_i^2 \tag{C.29}$$

Et lors de la mise à jour du poids w_i^t à l'itération $t+1$:

$$w_i^{t+1} = w_i^t - \epsilon_t \left[\frac{\partial E}{\partial w_i} - 2\lambda w_i^t \right] \tag{C.30}$$

Le coefficient λ peut décroître avec le nombre d'itérations et est en général de l'ordre de $0,01$ pour un apprentissage avec gradient global, plus faible pour un apprentissage avec gradient stochastique.

C.7 Sélection de connexions

Ce paragraphe présente un algorithme de sélection de l'architecture d'un réseau de neurones proposé par Cottrel et Al. dans [Cottrel1995]. La méthode est applicable à tout réseau de neurones mais n'a été démontrée que pour la classe de réseau de neurones utilisée pour la régression (paragraphe C.2, page 198). Les propriétés qui suivent ne sont vraies que des réseaux à une couche cachée et dont les sorties sont linéaires. Soit (X_k, Y_k) un exemple de la base d'apprentissage, les résidus de la régression sont supposés normaux et i.i.d. L'erreur est donc (voir paragraphe C.2.1) :

$$e\left(W, X_k, Y_k\right) = \left(f\left(W, X_k\right) - Y_k\right)^2$$

On peut estimer la loi asymptotique des coefficients du réseau de neurones. Des connexions ayant un rôle peu important peuvent alors être supprimées sans nuire à l'apprentissage en testant la nullité du coefficient associé. On note $\widehat{W}$ les poids trouvés par apprentissage et $\overset{*}{W}$ les poids optimaux. On définit :

$$\text{la suite } \widehat{\varepsilon_k} = f\left(\widehat{W}, X_k\right) - Y_k,\ \widehat{\sigma}_N^2 = \frac{1}{N}\sum_{k=1}^{N} \widehat{\varepsilon_k}^2 \tag{C.31}$$

$$\text{la matrice } \widehat{\Sigma_N} = \frac{1}{N}\left[\nabla_{\widehat{W}} e\left(W, X_k, Y_k\right)\right]\left[\nabla_{\widehat{W}} e\left(W, X_k, Y_k\right)\right]' \tag{C.32}$$

Théorème C.7.1 : loi asymptotique des coefficients
Soit f un réseau de neurone défini par (C.1.3) composé de :
– une couche d'entrées
– une couche cachée dont les fonctions de transfert sont sigmoïdes
– une couche de sortie dont les fonctions de transfert sont linéaires
Ce réseau sert de modèle pour la fonction f dans le problème C.1.6 avec un échantillon $\left(\left(X_1, Y_1\right), \ldots, \left(X_N, Y_N\right)\right)$, les résidus sont supposés normaux.
La suite $(\widehat{\epsilon_k})$ définie par (C.31) vérifie :

$$\frac{1}{N}\sum_{i=1}^{N} \widehat{\epsilon_k} = 0 = \mathbb{E}\left(f\left(\widehat{W}, X\right) - Y\right)$$

Et le vecteur aléatoire $\widehat{W} - W^*$ vérifie :

$$\sqrt{N}\left[\widehat{W} - W^*\right] \overset{T\to+\infty}{\longrightarrow} \mathcal{N}\left(0, \widehat{\sigma_N}^2 \widehat{\Sigma_N}\right)$$

Où la matrice $\widehat{\Sigma_N}$ est définie par (C.32).

La démonstration de ce théorème est donnée par l'article [Cottrel1995]. Ce théorème mène au corollaire

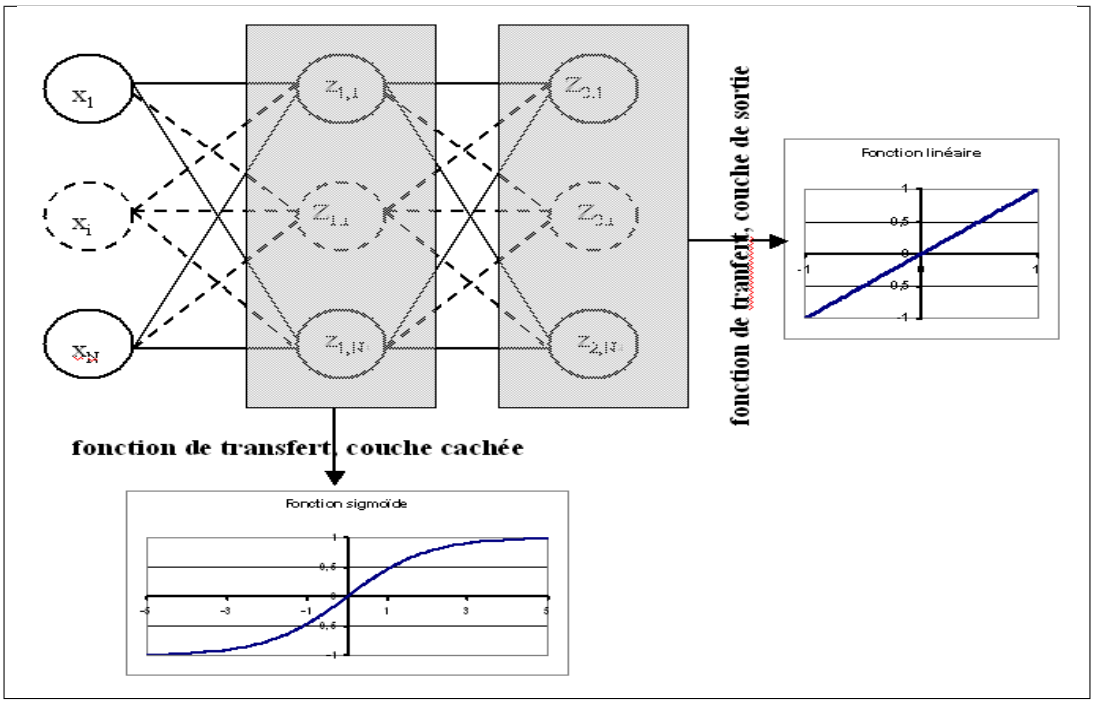


Fig. C.15: Réseau de neurones pour lequel la sélection de connexions s'applique.

suivant :

Corollaire C.7.2 : nullité d'un coefficient
Les notations utilisées sont celles du théorème C.7.1. Soit w_k un poids du réseau de neurones d'indice quelconque k. Sa valeur estimée est $\widehat{w_k}$, sa valeur optimale w_k^*. D'après le théorème C.7.1 :

$$N \frac{(\widehat{w_k} - w_k^*)^2}{\widehat{\sigma_N}^2 \left(\widehat{\Sigma_N}^{-1}\right)_{kk}} \overset{T \to +\infty}{\longrightarrow} \chi_1^2$$

Ce résultat permet, à partir d'un réseau de neurones, de supprimer les connexions pour lesquelles l'hypothèse de nullité n'est pas réfutée. Afin d'aboutir à l'architecture minimale adaptée au problème, Cottrel

et Al. proposent dans [Cottrel1995] l'algorithme suivant :

Algorithme C.7.3 : sélection d'architecture
Les notations utilisées sont celles du théorème C.7.1. f est un réseau de neurones de paramètres W. On définit la constante τ, en général $\tau = 3,84$ puisque $\mathbb{P}(X < \tau) = 0,95$ si $X \sim \chi^2_1$.

Etape A : initialisation
Une architecture est choisie pour le réseau de neurones f incluant un nombre M de paramètres.

Etape B : apprentissage
Le réseau de neurones f est appris. On calcule les nombre et matrice $\widehat{\sigma_N}^2$ et $\widehat{\Sigma_N}$. La base d'apprentissage contient N exemples.

Etape C : test
pour $k = 1$ **à** M **faire**

$$t_k \longleftarrow N \frac{\widehat{w_k}^2}{\widehat{\sigma_N}^2 \left(\widehat{\Sigma_N}^{-1}\right)_{kk}}$$

fin pour

Etape D : sélection
$k' \longleftarrow \arg\min_k t_k$
si $t_{k'} < \tau$ **alors**
Le modèle obtenu est supposé être le modèle optimal. L'algorithme s'arrête.
sinon
La connexion k' est supprimée ou le poids $w_{k'}$ est maintenue à zéro.
$M \longleftarrow M - 1$
Retour à l'étape B.
fin si

Remarque C.7.4: minimum local
Cet algorithme est sensible au minimum local trouvé lors de l'apprentissage, il est préférable d'utiliser des méthodes du second ordre afin d'assurer une meilleure convergence du réseau de neurones.

Remarque C.7.5: suppression de plusieurs connexions simultanément
L'étape D ne supprime qu'une seule connexion. Comme l'étape B est coûteuse en calcul, il peut être intéressant de supprimer toutes les connexions k qui vérifient $t_k < \tau$. Il est toutefois conseillé de ne pas enlever trop de connexions simultanément puisque la suppression d'une connexion nulle peut réhausser le test d'une autre connexion, nulle à cette même itération, mais non nulle à l'itération suivante.

Remarque C.7.6: minimum local
Dans l'article [Cottrel1995], les auteurs valident leur algorithme dans le cas d'une régression grâce à

l'algorithme C.7.7.

Algorithme C.7.7 : validation de l'algorithme C.7.3
Etape A : choix aléatoire d'un modèle

1. Un réseau de neurones est choisi aléatoirement, soit $f : \mathbb{R}^p \longrightarrow \mathbb{R}$ la fonction qu'il représente.
2. Une base d'apprentissage A (ou échantillon) de N observations est générée aléatoirement à partir de ce modèle :

$$\begin{array}{l}\text{soit } (\epsilon_i)_{1 \leqslant i \leqslant N} \text{ un bruit blanc (voir définition C.1.5)} \\ A = \left\{ (X_i, Y_i)_{1 \leqslant i \leqslant N} \middle| \, \forall i \in \{1, \cdots, N\}, \; Y_i = f(X_i) + \epsilon_i \right\}\end{array}$$

Etape B : choix aléatoire d'un modèle
L'algorithme C.7.3 à un réseau de neurone plus riche que le modèle choisi dans l'étape A. Le modèle sélectionné est noté g.

Etape C : validation
Si $\|f - g\| \approx 0$, l'algorithme C.7.3 est validé.

C.8 Analyse en composantes principales (ACP)

Cet algorithme est proposé dans [Song1997].

C.8.1 Principe

L'algorithme implémentant l'analyse en composantes principales est basé sur un réseau linéaire dit "diabolo", ce réseau possède une couche d'entrées à N entrées, une couche cachée et une couche de sortie à N sorties. L'objectif est d'apprendre la fonction identité sur l'espace $\mathbb{R}^N$. Ce ne sont plus les sorties qui nous intéressent mais la couche cachée intermédiaire qui effectue une compression ou projection des vecteurs d'entrées puisque les entrées et les sorties du réseau auront pour but d'être identiques (voir figure C.16). La figure C.17 illustre un exemple de compression de vecteur de $\mathbb{R}^3$ dans $\mathbb{R}^2$.

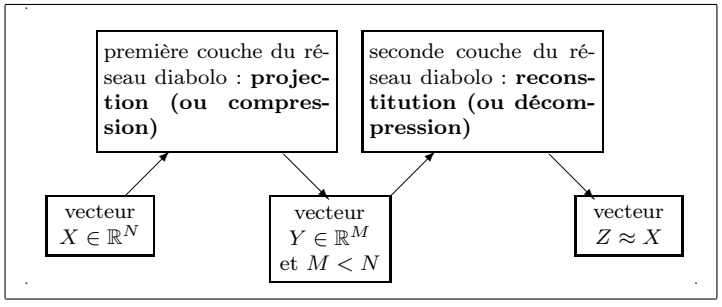


Fig. C.16: Principe de la compression par un réseau diabolo

La compression et décompression ne sont pas inverses l'une de l'autre, à moins que l'erreur (C.33) soit nulle. La décompression s'effectue donc avec des pertes d'information. L'enjeu de l'ACP est de trouver un bon compromis entre le nombre de coefficients et la perte d'information tôlérée. Dans le cas de l'ACP, la compression est "linéaire", c'est une projection.

Le réseau suivant possède 3 entrées et 3 sorties. Minimiser l'erreur :

$$\sum_{k=1}^{N} E\left(X_k, X_k\right)$$

revient à compresser un vecteur de dimension 3 en un vecteur de dimension 2. Les coefficients de la première couche du réseau de neurones permettent de compresser les données. Les coefficients de la seconde couche permettent de les décompresser.

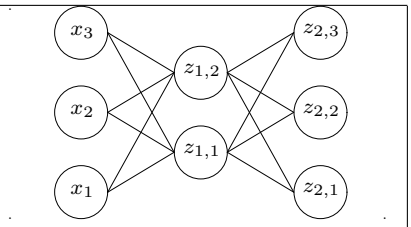


Fig. C.17: Réseau diabolo : réduction d'une dimension

C.8.2 Problème de l'analyse en composantes principales

L'analyse en composantes principales ou ACP est définie de la manière suivante :

Problème C.8.1 : analyse en composantes principales (ACP)
Soit $(X_i)_{1 \leqslant i \leqslant N}$ avec $\forall i \in \{1, \ldots, N\},\ X_i \in \mathbb{R}^p$.
Soit $W \in M_{p,d}(\mathbb{R})$, $W = (C_1, \ldots, C_d)$ où les vecteurs (C_i) sont les colonnes de W et $d < p$. On suppose également que les (C_i) forment une base othonormée. Par conséquent :

$$W'W = I_d$$

$(W'X_i)_{1 \leqslant i \leqslant N}$ est l'ensemble des vecteurs (X_i) projetés sur le sous-espace vectoriel engendré par les vecteurs (C_i).
Réaliser une analyse en composantes principales, c'est trouver le meilleur plan de projection pour les vecteurs (X_i), celui qui maximise l'inertie de ce nuage de points, c'est donc trouver W^* tel que :

$$W^* = \underset{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}}{\arg\max}\ E(W) = \underset{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}}{\arg\max} \left[\sum_{i=1}^{N} \left\| W'X_i \right\|^2 \right] \tag{C.33}$$

Le terme $E(W)$ est l'inertie du nuage de points (X_i) projeté sur le sous-espace vectoriel défini par les vecteurs colonnes de la matrice W.

C.8.3 Résolution d'une ACP avec un réseau de neurones diabolo

Un théorème est nécessaire avant de construire le réseau de neurones menant à la résolution du problème C.8.1 afin de passer d'une optimisation sous contrainte à une optimisation sans contrainte.

Théorème C.8.2 : résolution de l'ACP
Les notations utilisées sont celles du problème C.8.1. Dans ce cas :

$$S = \underset{\substack{W\in M_{p,d}(\mathbb{R})\\ W'W=I_d}}{\arg\max} \left[\sum_{i=1}^{N} \left\|W'X_i\right\|^2\right] = \underset{W\in M_{p,d}(\mathbb{R})}{\arg\min} \left[\sum_{i=1}^{N} \left\|WW'X_i - X_i\right\|^2\right] \tag{C.34}$$

De plus S est l'espace vectoriel engendré par les d vecteurs propres de la matrice $XX' = \sum_{i=1}^{N} X_i X_i'$ associés aux d valeurs propres de plus grand module.

Démonstration (théorème C.8.2) :

Partie A (démonstration de C.8.2)

L'objectif de cette partie est de chercher la valeur de :

$$\max_{\substack{W\in M_{p,d}(\mathbb{R})\\ W'W=I_d}} E(W)$$

Soit $X = (X_1, \ldots, X_N) \in (\mathbb{R}^p)^N$, alors :

$$E(W) = \sum_{i=1}^{N} \left\|W'X_i\right\|^2 = tr\left(X'WW'X\right) = tr\left(XX'WW'\right)$$

La matrice XX' est symétrique, elle est donc diagonalisable et il existe une matrice $P \in M_p(\mathbb{R})$ telle que :

$$\begin{array}{l} P'XX'P = D_X \text{ avec } D_X \text{ diagonale} \\ P'P = I_p \end{array} \tag{C.35}$$

Soit $P = (P_1, \ldots, P_p)$ les vecteurs propres de la matrice XX' associés aux valeurs propres $(\lambda_1, \ldots, \lambda_p)$ telles que $|\lambda_1| \geqslant ... \geqslant |\lambda_p|$. Pour mémoire, $W = (C_1, \ldots, C_d)$, et on a :

$$\begin{array}{l} \forall i \in \{1, \ldots, p\},\ XX'P_i = \lambda_i P_i \\ \forall i \in \{1, \ldots, d\},\ C_i = P_i \Longrightarrow XX'WW' = D_{X,d} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_d \end{pmatrix} \end{array}$$

D'où :

$$E(W) = tr\left(XX'WW'\right) = tr\left(PD_XP'WW'\right) = tr\left(D_XP'WW'P\right)$$

Donc :

$$\max_{\substack{W\in M_{p,d}(\mathbb{R})\\ W'W=I_d}} E(W) = \max_{\substack{W\in M_{p,d}(\mathbb{R})\\ W'W=I_d}} tr\left(D_X P'WW'P\right) = \max_{\substack{Y\in M_{p,d}(\mathbb{R})\\ Y'Y=I_d}} tr\left(D_X YY'\right) = \sum_{i=1}^{d}\lambda_i \tag{C.36}$$

Partie B (démonstration de C.8.2)

Soit $Y \in \max_{\substack{W\in M_{p,d}(\mathbb{R})\\ W'W=I_d}} tr\left(X'WW'X\right)$, $Y = (Y_1,\ldots,Y_d) = \left(y_i^k\right)_{\substack{1\leqslant i\leqslant d\\ 1\leqslant k\leqslant p}}$.

Chaque vecteur Y_i est écrit dans la base $(P_1,\ldots,P_p)$ définie en (C.35) :

$$\forall i \in \{1,\ldots,d\},\ Y_i = \sum_{k=1}^{p} y_i^k P_p$$

Comme $Y'Y = I_d$, les vecteurs $(Y_1,\ldots,Y_d)$ sont orthogonaux deux à deux et normés, ils vérifient donc :

$$\begin{cases} \forall i \in \{1,\ldots,d\}, & \sum\limits_{k=1}^{p}\left(y_i^k\right)^2 = 1 \\ \forall (i,j) \in \{1,\ldots,d\}^2, & \sum\limits_{k=1}^{p} y_i^k y_j^k = 0 \end{cases}$$

De plus :

$$XX'YY' = XX'\left(\sum_{i=1}^{d} Y_i Y_i'\right) = \sum_{i=1}^{d} XX'Y_iY_i'$$

On en déduit que :

$$\begin{aligned} \forall i \in \{1,\ldots,d\},\ XX'Y_iY_i' &= XX'\left(\sum_{k=1}^{p} y_i^k P_k\right)\left(\sum_{k=1}^{p} y_i^k P_k\right)' \\ &= \left(\sum_{k=1}^{p}\lambda_k y_i^k P_k\right)\left(\sum_{k=1}^{p} y_i^k P_k\right)' \end{aligned}$$

D'où :

$$\forall i \in \{1,\ldots,d\},\ tr\left(XX'Y_iY_i'\right) = \sum_{k=1}^{p}\lambda_k\left(y_i^k\right)^2$$

Et :

$$\begin{aligned} tr\left(XX'YY'\right) &= \sum_{i=1}^{d}\sum_{k=1}^{p}\lambda_k\left(y_i^k\right)^2 \\ tr\left(XX'YY'\right) &= \sum_{k=1}^{p}\lambda_k\left(\sum_{i=1}^{d}\left(y_i^k\right)^2\right) = \sum_{k=1}^{p}\lambda_k \end{aligned}$$

Ceci permet d'affirmer que :

$$Y \in \max_{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}} tr\left(X'WW'X\right) \Longrightarrow vect\left(Y_1, \ldots, Y_d\right) = vect\left(P_1, \ldots, P_d\right) \tag{C.37}$$

Les équations (C.36) et (C.37) démontrent la seconde partie du théorème C.8.2.

Partie C (démonstration de C.8.2)

$$\begin{aligned}
\sum_{i=1}^{n} \left\|WW'X_i - X_i\right\|^2 &= \sum_{i=1}^{n} \left\|\left(WW' - I_N\right) X_i\right\|^2 \\
&= tr\left(X'\left(WW' - I_p\right)^2 X\right) \\
&= tr\left(XX'\left(\left(WW'\right)^2 - 2WW' + I_p\right)\right) \\
&= tr\left(XX'\left(WW'WW' - 2WW' + I_p\right)\right) \\
&= tr\left(XX'\left(-WW' + I_p\right)\right) \\
&= -tr\left(XX'WW'\right) + tr\left(XX'\right)
\end{aligned}$$

D'où :

$$\max_{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}} \left(\sum_{i=1}^{N} \left\|W'X_i\right\|^2\right) = \min_{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}} \left(\sum_{i=1}^{N} \left\|WW'X_i - X_i\right\|^2\right) \tag{C.38}$$

Partie D (démonstration de C.8.2)

XX' est une matrice symétrique, elle est donc diagonalisable :

$$\exists P \in GL_N\left(\mathbb{R}\right) \text{ telle que } P'XX'P = D_p \text{ où } D_p \text{ est diagonale}$$

On en déduit que :

$$\begin{aligned}
\sum_{i=1}^{N} \left\|WW'X_i - X_i\right\|^2 &= tr\left(XX'\left(WW' - I_p\right)^2\right) \\
&= tr\left(PP'XX'PP'\left(WW' - I_p\right)^2\right) \\
&= tr\left(PD_pP'\left(WW' - I_p\right)^2\right) \\
&= tr\left(D_p\left(P'WW'P - I_p\right)^2\right) \\
&= tr\left(D_p\left(YY' - I_p\right)^2\right) \text{ avec } Y = P'W
\end{aligned}$$

D'où :

$$\arg\min_{Y} \left\{ tr\left(D_p \left(YY' - I_p \right)^2 \right) \right\} = \left\{ Y \in M_{Nd}(\mathbb{R}) \,|\, YY' = I_d \right\} \tag{C.39}$$

Finalement, l'équation (C.39) permet de démontrer la première partie du théorème C.8.2, à savoir (C.34) :

$$S = \underset{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}}{\arg\max} \left[\sum_{i=1}^{N} \left\| W'X_i \right\|^2 \right] = \underset{W \in M_{p,d}(\mathbb{R})}{\arg\min} \left[\sum_{i=1}^{N} \left\| WW'X_i - X_i \right\|^2 \right]$$

(C.8.2) □

C.8.4 Calcul de valeurs propres et de vecteurs propres

Le calcul des valeurs propres et des vecteurs propres d'une matrice fait intervenir un réseau diabolo composé d'une seule couche cachée et d'une couche de sortie avec des fonctions de transfert linéaires. On note sous forme de matrice (W) les coefficients de la seconde couche du réseau dont les biais sont nuls. On note d le nombre de neurones sur la couche cachée, et p le nombre d'entrées.

$$\forall i \in \{1, ..., d\}, \; y_{1,i} = \sum_{j=1}^{p} w_{ji} x_j$$

Soit $X \in \mathbb{R}^p$ les entrées, $Y = (y_{1,1}, ..., y_{1,d}) \in \mathbb{R}^d$, on obtient que : $Y = W'X$.

Les poids de la seconde couche sont définis comme suit :

$$\forall (i,j) \in \{1, ..., p\} \times \{1, ..., d\} \; w_{2,j,i} = w_{1,i,j}$$

Par conséquent, le vecteur des sorties $Z \in \mathbb{R}^p$ du réseau ainsi construit est : $Z = WW'X$

On veut minimiser l'erreur pour $(X_i)_{1 \leqslant i \leqslant N}$:

$$E = \sum_{i=1}^{N} \left\| WW'X_i - X_i \right\|^2$$

Il suffit d'apprendre le réseau de neurones pour obtenir :

$$W_d^* = \underset{W \in M_{pd}(\mathbb{R})}{\arg\max} \sum_{i=1}^{N} \left\| WW'X_i - X_i \right\|^2$$

D'après ce qui précède, l'espace engendré par les vecteurs colonnes de W est l'espace engendré par les k premiers vecteurs propres de la matrice $XX' = (X_1, ..., X_P)(X_1, ..., X_P)'$ associés aux k premières valeurs propres classées par ordre décroissant de module.

On en déduit que W_1^* est le vecteur propre de la matrice M associée à la valeur propre de plus grand module. W_2^* est l'espace engendré par les deux premiers vecteurs. Grâce à une orthonormalisation de

Schmidt (voir définition C.8.3), on en déduit à partir de W_1^* et W_2^*, les deux premiers vecteurs propres. Par récurrence, on trouve l'ensemble des vecteurs propres de la matrice XX'.

Définition C.8.3 : orthonormalisation de Schmidt
L'orthonormalisation de Shmidt :
Soit $(e_i)_{1\leqslant i\leqslant N}$ une base de $\mathbb{R}^p$
On définit la famille $(\varepsilon_i)_{1\leqslant i\leqslant p}$ par :

$$\varepsilon_1 = \frac{e_1}{\|e_1\|}$$

$$\forall i \in \{1,\cdots,p\},\ \varepsilon_i = \frac{e_i - \sum_{j=1}^{i-1} < e_i, \varepsilon_j > \varepsilon_j}{\left\| e_i - \sum_{j=1}^{i-1} < e_i, \varepsilon_j > \varepsilon_j \right\|}$$

Remarque C.8.4: dénominateur non nul
$e_i - \sum_{j=1}^{i-1} < e_i, \varepsilon_j > \varepsilon_j \neq 0$ car $\forall k \in \{1,...,N\},\ vect\,(e_1,...,e_k) = vect\,(\varepsilon_1,...,\varepsilon_k)$

Propriété C.8.5 : base orthonormée
La famille $(\varepsilon_i)_{1\leqslant i\leqslant p}$ est une base orthonormée de $\mathbb{R}^p$

L'algorithme qui permet de déterminer les vecteurs propres de la matrice XX' définie par le théorème C.8.2 est le suivant :

Algorithme C.8.6 : vecteurs propres
Les notations utilisées sont celles du théorème C.8.2. On note V_d^* la matrice des d vecteurs propres de la matrice XX' associés aux d valeurs propres de plus grands module.
pour $d = 1$ **à** p **faire**
  Un réseau diabolo est construit avec les poids $W_d \in M_{p,d}(\mathbb{R})$ puis appris. Le résultat de cet apprentissage sont les poids W_d^*.
  si $d > 1$ **alors**
    L'orthonormalisation de Schmit permet de déduire V_d^* de V_{d-1}^* et W_d^*.
  sinon
    $V_d^* = W_d^*$
  fin si
fin pour

C.8.5 Analyse en Composantes Principales (ACP)

L'analyse en composantes principales permet d'analyser une liste d'individus décrits par des variables. Comme exemple, il suffit de prendre les informations extraites du recensement de la population française qui permet de décrire chaque habitant par des variables telles que la catégorie socio-professionnelle, la salaire ou le niveau d'étude.

Soit $(X_1,...,X_N)$ un ensemble de N individus décrits par p variables :

$$\forall i \in \{1,...,N\},\ X_i \in \mathbb{R}^p$$

L'ACP consiste à projeter ce nuage de point sur un plan qui conserve le maximum d'information. Par conséquent, il s'agit de résoudre le problème :

$$W^* = \underset{\substack{W \in M_{p,d}(\mathbb{R}) \\ W'W = I_d}}{\arg\min} \left(\sum_{i=1}^{N} \left\| W' X_i \right\|^2 \right) \text{ avec } d < N$$

Ce problème a été résolu dans les paragraphes C.8.2 et C.8.4, il suffit d'appliquer l'algorithme C.8.6.

Remarque C.8.7: expérience

Soit $(X_i)_{1 \leqslant i \leqslant N}$ avec $\forall i \in \{1, ..., N\}$, $X_i \in \mathbb{R}^p$. Soit $(P_1, \ldots, P_p)$ l'ensemble des vecteurs propres normés de la matrice XX' associés aux valeurs propres $(\lambda_1, \ldots, \lambda_p)$ classées par ordre décroissant de modules. On définit $\forall d \in \{1, \cdots, p\}$, $W_d = (P_1, \ldots, P_d) \in M_{p,d}$. On définit alors l'inertie I_d du nuage de points projeté sur l'espace vectoriel défini par P_d. On suppose que le nuage de points est centré, alors :

$$\forall d \in \{1, \cdots, p\}, \; I_d = \sum_{k=1}^{N} \left(P_d' X_k \right)^2 = tr\left(X' P_d P_d' X \right) = tr\left(X X' P_d P_d' \right) = \lambda_d$$

Comme $(P_1, \ldots, P_p)$ est une base orthonormée de $\mathbb{R}^p$, on en déduit que :

$$I = \sum_{k=1}^{P} X_k' X_k = \sum_{d=1}^{N} I_d = \sum_{d=1}^{p} \lambda_d$$

De manière empirique, on observe fréquemment que la courbe $(d, I_d)_{1 \leqslant d \leqslant p}$ montre un point d'inflexion (figure C.18). Dans cet exemple, le point d'inflexion correspond à $d = 4$. En analyse des données, on considère empiriquement que seuls les quatres premières dimensions contiennent de l'information.

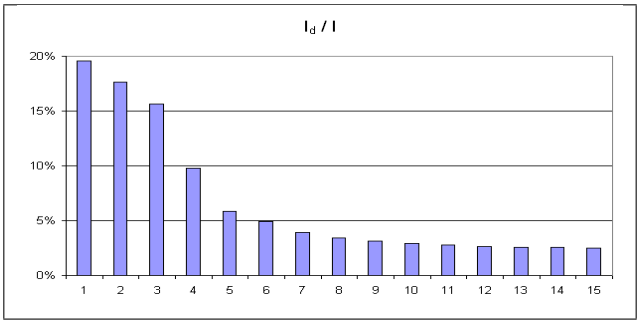


Fig. C.18: Courbe d'inertie : point d'inflexion pour $d = 4$, l'expérience montre que généralement, seules les projections sur un ou plusieurs des quatre premiers vecteurs propres reflètera l'information contenue par le nuage de points.

Annexe D

Support Vector Machines

Les *Support Vector Machine* ou *Séparateurs à Vastes Marges* (SVM) ont été pour la première fois présentés par V. Vapnik dès 1979 (voir [Vapnik1979]) et sont plus amplement développés dans [Vapnik1998]. Les définitions et résultats proposés sont extraits de [Burges1998], document plus didactique d'après son auteur, [Smola2004] - cet article existe en une version plus étendue (voir [Smola1998]) - document plus complet qui présente la régression à partir de SVM et l'article [Müller2001], document plus récent qui évoque notamment l'analyse en composantes principales à partir de SVM. Ce dernier document applique les SVM à la reconnaissance de caractères. Cette annexe n'a pas pour but de décrire en détail ces modèles mais seulement de les introduire sommairement. Le site internet *http ://www.kernel-machines.org/* référence tous ces documents et recense les derniers développements autour des méthodes à noyaux dont font partie les SVM. Il référence également un large panel d'applications ou de code informatique permettant d'utiliser les méthodes à noyaux.

D.1 Séparateur linéaire

D.1.1 Ensemble séparable

On s'intéresse tout d'abord à l'hyperplan séparateur d'un ensemble de points répartis en deux classes. Cet ensemble est noté $(X_i, Y_i)_{1 \leqslant i \leqslant N}$ où, $\forall i,\ X_i \in \mathbb{R}^d$ et $Y_i \in \{-1, 1\}$. Pour simplifier les expressions par la suite, les deux classes sont donc labellées -1 et 1. On cherche alors un vecteur w et une constante b qui vérifient :

$$\forall i,\ 1 \leqslant i \leqslant N,\ Y_i = \begin{cases} -1 & \text{si } w.X_i + b \geqslant 1 \\ 1 & \text{si } w.X_i + b \leqslant -1 \end{cases}$$

On cherche donc w et b tels que :

$$\forall i,\ 1 \leqslant i \leqslant N,\ Y_i\,(w.X_i + b) - 1 \geqslant 0$$

Comme on cherche également un vecteur w de norme minimum, l'hyperplan cherché est la solution du

problème de minimsation suivant :

Problème D.1.1 : meilleur hyperplan séparateur, cas séparable
Le meilleur hyperplan séparateur de l'ensemble de points labellés $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ est la solution d'un problème de minimisation. Cet hyperplan a pour équation $x.w^* + b^* = 0$ où w^* et b^* vérifient :

$$\begin{aligned}(w^*, b^*) &= \underset{w,b}{\arg\min} \tfrac{1}{2} \|w\|^2 \\ &\quad \text{avec } \forall i,\ Y_i \left(X_i.w + b\right) - 1 \geqslant 0\end{aligned}$$

La résolution d'un tel problème s'effectue à l'aide des multiplicateurs de Lagrange, on affecte à chaque contrainte le coefficient α_i, il s'agit alors de minimiser l'expression :

$$L_P = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i Y_i \left(X_i.w + b\right) + \sum_{i=1}^{N} \alpha_i \tag{D.1}$$

En dérivant par rapport à w et b, on obtient que :

$$w = \sum_{i=1}^{N} \alpha_i Y_i X_i \tag{D.2}$$

$$\sum_{i=1}^{N} \alpha_i Y_i = 0 \tag{D.3}$$

Par conséquent, on peut substituer l'expression D.1 par :

$$L_D = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, Y_i Y_j \, X_i.X_j - \sum_{i=1}^{N} \alpha_i \tag{D.4}$$

Cette dernière équation (D.4) est importante puisqu'elle permet d'introduire les SVM non linéaires pour lesquels le produit scalaire $X_i.X_j$ sera remplacé par une fonction noyau $K\left(X_i, X_j\right)$.

D.1.2 Ensemble non séparable

Le paragraphe précédent supposait que l'ensemble $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ était séparable ce qui, d'après le paragraphe D.2.2 implique dans la plupart des cas que $N \leqslant d + 1$. Pour un ensemble non séparable (voir figure D.1), il est impossible de trouver un hyperplan séparateur. Par conséquent, il n'existe pas de solution au problème D.1.1 vérifiant les contraintes telles qu'elles ont été exprimées. La recherche du meilleur hyperplan séparateur devient alors l'énoncé D.1.2.

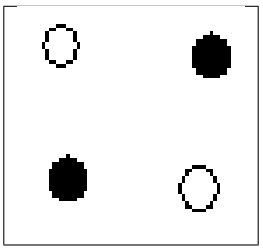


Fig. D.1: Exemple d'un nuage de points non séparable par un hyperplan.

Problème D.1.2 : meilleur hyperplan séparateur, cas non séparable
Soit $C \in \mathbb{R}_+^*$ une constante et $k \in \mathbb{N}^*$ un entier, le meilleur hyperplan séparateur de l'ensemble de points labellés $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ est la solution d'un problème de minimisation. Cet hyperplan a pour équation $x.w^* + b^* = 0$ où w^* et b^* vérifient :

$$\begin{array}{rcl} (w^*, b^*) & = & \underset{w,b}{\arg\min} \dfrac{1}{2} \|w\|^2 + C \left(\sum\limits_{i=1}^{N} \xi_i \right)^k \\ \text{avec} & & \forall i,\ Y_i \left(X_i.w + b + \xi_i \right) - 1 \geqslant 0 \\ \text{et} & & \forall i,\ \xi_i \geqslant 0 \end{array}$$

C et k sont des constantes à déterminer. Toutefois, dans le cas où $k = 1$, la solution du problème précédent est identique à celle du problème suivant :

Problème D.1.3 : meilleur hyperplan séparateur, cas non séparable, problème dual
Soit $C \in \mathbb{R}_+^*$ une constante, le meilleur hyperplan séparateur de l'ensemble de points labellés $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ est la solution d'un problème de minimisation.

$$\begin{array}{rcl} (\alpha_i^*) & = & \underset{\alpha_i}{\arg\min} \dfrac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} \alpha_i \alpha_j\, Y_i Y_j\, X_i.X_j - \sum\limits_{i=1}^{N} \alpha_i \\ \text{avec} & & \forall i,\ 1 \leqslant \alpha_i \leqslant C \\ \text{et} & & \sum\limits_{i=1}^{N} Y_i\, \alpha_i = 0 \end{array}$$

L'hyperplan séparateur est donné par l'équation $x.w + b = 0$ où $w = \sum\limits_{i=1}^{N} \alpha_i Y_i X_i$.

Ce dernier problème est appelée la forme duale du problème D.1.2.

D.2 Dimension de Vapnik-Chervonenkis (VC)

D.2.1 Définition

Dans le problème de classification introduit au chapitre D.1, la dimension de Vapnik-Chervonenkis sert à majorer le risque d'erreur de classification empirique au risque d'erreur théorique. Nous allons tout d'abord définir la dimension de Vapnik-Chervonenkis pour un ensemble de points donné et noté $(X_i)_{1 \leqslant i \leqslant N}$ et une

classe de fonction $f(x,\alpha)$ paramétrée par α.

Définition D.2.1 : dimension de Vapnik-Chervonenkis
Soit $(X_i)_{1\leqslant i\leqslant N}$ un ensemble de points appartenant à $\mathbb{R}^d$. On définit une fonction $f(x,\alpha)$: $\mathbb{R}^d \times \Omega \longmapsto \mathbb{R}$ où $x \in \mathbb{R}^d$ et $\alpha \in \Omega$. Ω est appelé l'ensemble des paramètres. On définit la dimension de Vapnik-Chervonenkis comme étant le nombre de suites $(Y_i)_{1\leqslant i\leqslant N} \in \{-1,1\}^N$ vérifiant :

$$\exists \alpha \in \Omega, \text{ tel que } \forall i,\ 1 \leqslant i \leqslant N,\ sgn\left(f\left(X_i,\alpha\right)\right) = Y_i$$

La fonction $sgn(x)$ désigne le signe de x : $sgn(x) = \left\{ \begin{array}{cc} 1 & \text{si } x \geqslant 0 \\ -1 & \text{si } x < 0 \end{array} \right.$
Par définition, cette dimension est inférieure à 2^N.

D.2.2 Résultats

Dans le cas où la fonction f est linéaire, il existe quelques résultats intéressants.

Théorème D.2.2 : dimension VC d'un ensemble de vecteurs linéairement indépendants
Soit un ensemble de N points inclus dans l'espace vectoriel $\mathbb{R}^d$ dont un définit l'origine. Alors les N points peuvent être séparés de n'importe quel manière en deux classes par des hyperplans orientés si et seulement si les vecteurs positions sont linéairement indépendants.

Corollaire D.2.3 : dimension VC d'un ensemble de vecteurs linéairement indépendants
La dimension de Vapnik-Chervonenkis d'un ensemble d'hyperplans séparateurs de $\mathbb{R}^d$ est $d+1$ puisqu'il est toujours possible de choisir $d+1$ points linéairement indépendants qui puissent être séparés quelque soit leurs classes.

D.2.3 Exemple

On définit la suite de points $(X_i)_{1\leqslant i\leqslant N}$ par $\forall i,\ 1 \leqslant i \leqslant N,\ X_i = 10^{-i}$ et l'ensemble de fonctions :

$$\left\{ \alpha \in \mathbb{R},\ f(x,\alpha) = \left\{ \begin{array}{cc} 1 & \text{si } \sin \alpha x \geqslant 0 \\ -1 & \text{si } \sin \alpha x < 0 \end{array} \right. \right\}$$

Quelque soit la suite $(Y_i)_{1\leqslant i\leqslant N} \in \{-1,1\}^N$, il est possible de choisir :

$$\alpha = \pi \left(1 + \sum_{i=1}^{N} \frac{(1-Y_i)\,10^i}{2} \right)$$

De telle sorte que : $\forall i,\ f(X_i,\alpha) = Y_i$. Par conséquent, la dimension VC cet ensemble de points associés à l'ensemble de fonctions f est 2^N.

D.2.4 Risque

On définit maintenant le risque théorique de classification comme étant :

$$R(\alpha) = \int \frac{1}{2} |y - f(x, \alpha)| \, dP(x, y) \tag{D.5}$$

Et le risque empirique pour le nuage de points $(X_i, Y_i)_{1 \leqslant i \leqslant N}$ par :

$$R_{emp}(\alpha) = \frac{1}{2N} \sum_{i=1}^{N} |Y_i - f(X_i, \alpha)| \tag{D.6}$$

Théorème D.2.4 : majoration du risque empirique
En reprenant les notations utilisées dans les expressions (D.5) et (D.6). Pour un nuage de points $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$, on démontre (voir [Vapnik1995]) que $\forall \eta \in [0, 1]$:

$$\mathbb{P}\left(R(\alpha) \leqslant R_{emp}(\alpha) + \sqrt{\frac{h\left(1 + \ln \frac{2N}{h}\right) - \ln \frac{\eta}{4}}{N}} \right) = 1 - \eta$$

où h est la dimension de Vapnik-Chervonenkis.

D.3 Séparateur non linéaire

D.3.1 Principe

Il est possible d'étendre les SVM au cas non linéaire à partir du problème D.1.2 d'après [Boser1992] en remplaçant le produit scalaire $X_i.X_j$ par une fonction noyau telle qu'une fonction gaussienne [1].

Problème D.3.1 : meilleur hyperplan, cas non séparable, non linéaire, problème dual
Soit $C \in \mathbb{R}_+^*$ une constante, soit $K : \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}^+$ une fonction noyau, le meilleur hyperplan séparateur de l'ensemble de points labellés $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ est la solution d'un problème de minimisation :

$$\begin{array}{rcl} (\alpha_i^*) & = & \arg\min_{\alpha_i} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, Y_i Y_j \, K(X_i, X_j) - \sum_{i=1}^{N} \alpha_i \\ \text{avec} & & \forall i, \; 1 \leqslant \alpha_i \leqslant C \\ \text{et} & & \sum_{i=1}^{N} Y_i \, \alpha_i = 0 \end{array}$$

La classification d'un élément $x \in \mathbb{R}^d$ dépend du signe de la fonction :

$$f(x) = \sum_{i=1}^{N} \alpha_i Y_i K(X_i, x) + b$$

1. $K(X, Y) = \exp\left(-\frac{\|X - Y\|^2}{2\sigma^2}\right)$

D.3.2 Interprétation, exemple

Le problème D.3.1 revient en fait à projeter l'élément $X_i \in \mathbb{R}^d$ dans un autre espace de dimension généralement supérieure $\mathbb{R}^{d'}$ dans lequel la séparation sera un hyperplan. Par exemple, on définit le noyau $K : \mathbb{R}^2 \times \mathbb{R}^2 \longmapsto \mathbb{R}^+$ par :

$$K(X_i, X_j) = (X_i.X_j)^2$$

On définit également la fonction $\Phi : \mathbb{R}^2 \longmapsto \mathbb{R}^3$ par :

$$\Phi(x_1, x_2) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

On vérifie alors que :

$$K(X_i, X_j) = \Phi(X_i).\Phi(X_j)$$

Plus généralement, pour qu'un noyau K corresponde à un produit scalaire dans un espace de dimension supérieure, il suffit qu'il vérifie la conditions de Mercer (voir [Vapnik1995]) :

$$\begin{array}{l} \text{Pour toute fonction } g \text{ telle que : } \int g(x)^2 dx < +\infty \text{ alors} \\ \int K(x,y)\, g(x)g(y)dxdy \geqslant 0 \end{array}$$

D.3.3 Autre formulation

Le problème D.3.1 peut être formulé d'une manière différente proche de celle du problème D.1.2.

Problème D.3.2 : meilleur hyperplan séparateur, cas non séparable, non linéaire
Soit $C \in \mathbb{R}_+^*$ une constante, soit $K : \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}^+$ une fonction noyau, le meilleur hyperplan séparateur de l'ensemble de points labellés $(X_i, Y_i)_{1 \leqslant i \leqslant N} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^N$ est la solution d'un problème de minimisation :

$$\begin{array}{rcl} (\alpha_i^*, \xi_i^*) & = & \underset{\alpha_i}{\arg\min} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j Y_i Y_j K(X_i, X_j) + C \left(\sum_{i=1}^{N} \xi_i \right)^k \\ \text{avec} & & \forall i,\ Y_i \left(\sum_{k=1}^{N} \alpha_k Y_k K(X_k, x) + b + \xi_i \right) - 1 \geqslant 0 \\ \text{et} & & \forall i,\ \xi_i \geqslant 0 \end{array}$$

La classification d'un élément $x \in \mathbb{R}^d$ dépend du signe de la fonction :

$$f(x) = \sum_{i=1}^{N} \alpha_i Y_i K(X_i, x) + b$$

D.4 Extensions

D.4.1 Classification en plusieurs classes

Jusqu'à présent, le seul problème évoqué concerne une classification en deux classes. Une classification en N classes est néanmoins possible selon deux stratégies. La première consiste à isoler une classe contre toutes les autres puis à procéder récusivement de cette manière jusqu'à finalement obtenir un problème de classification en deux classes. La seconde stratégie consiste à regrouper le nombre de classes en deux groupes puis à appliquer la méthode des SVM. Ensuite, à l'intérieur de chaque groupe, on réitère cette méthode de manière à diviser le nombre de classes jusqu'à obtenir un problème de classification à deux classes.

Il existe une autre possibilité plus coûteuse et plus fiable. Si on désire réaliser une classification en N classes, plutôt que de réaliser au plus $N-1$ classifications en deux classes, on réalise $\frac{(N-1)^2}{2}$ classifications pour tous les couples de deux différentes classes. Il suffit de prendre la classe qui ressort le plus souvent vainqueur.

D.4.2 Ensembles réduits

L'article [Burges1997] propose une méthode permettant de réduire l'ensemble de point (X_i) afin d'accélerer la résolution du problème de minimisation, tout en n'accroissant que légèrement l'erreur ($\sim$1% d'après les auteurs).

L'article [Zhan2005] propose une autre méthode qui supprime des points. Plusieurs optimisations sont réalisées et, après chaque étape, les points proches des zones de forte courbure de la frontière sont enlevés de l'ensemble d'apprentissage. L'article étudie la perte de performances en fonction du nombre de points supprimés.

D.4.3 Sélection des paramètres

Les problèmes de minimisation D.1.2, D.1.3, D.3.1 et D.3.2 mentionnent une constante C dont l'article [Mattera1999] (voir également [Cherkassky2004]) discute le choix dans le cas non pas d'un problème de classification mais dans celui d'une régression à l'aide des SVM.

D.4.4 Régression

Annexe E

Modèles de Markov cachés

Cette annexe détaille les concepts et les propriétés des modèles de Markov cachés en évitant aussi souvent que possible les références à la reconnaissance de l'écriture manuscrite.

E.1 Chaîne de Markov

E.1.1 Définition

Une chaîne de Markov est un modèle probabiliste modélisant des séquences de symboles appartenant à un ensemble fini. Ces séquences peuvent être considérées également comme des suites entières finies.

Définition E.1.1 : chaîne de Markov
Soit M une chaîne de Markov.
Soit $Q = \{1, \cdots, N\}$ l'ensemble des états.
Soit $S = \bigcup_{T=1}^{+\infty} Q^T$ l'espace des séquences.
On note $s = (q_1, \ldots, q_{T_s}) \in S$ une séquence de longueur T_s.
Une chaîne de Markov est un modèle probabiliste sur S vérifiant les deux hypothèses suivantes :

1. L'état à l'instant t ne dépend que de l'état à l'instant $t-1$:

$$\forall s \in S,\ \forall t \in \{2, \cdots, T_s\},\ \mathbb{P}(q_t \mid q_1, \ldots, q_{t-1}, M) = \mathbb{P}(q_t \mid q_{t-1}, M)$$

   On appelle $\mathbb{P}(q_t \mid q_{t-1}, M)$ la *probabilité de transition* de l'état q_{t-1} à l'état q_t à l'instant t.

2. Les probabilités de transition ne dépendent pas du temps :

$$\forall s \in S,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}(q_t \mid q_{t-1}, M) = \mathbb{P}(q_{t'} \mid q_{t'-1}, M)$$

Afin de simplifier les notations ultérieurement, on définit pour la chaîne de Markov M, la matrice des probabilités de transition $A_M \in M_N(\mathbb{R})$:

$$A_M = (a_{M,ij})_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant N}} = \left(\mathbb{P}(q_t = j \mid q_{t-1} = i, M)\right)_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant N}} \tag{E.1}$$

On définit également le vecteur des probabilités d'entrées $\pi_M \in \mathbb{R}^N$:

$$\forall i \in \{1, \ldots, N\}\,,\ \pi_{M,i} = (q_1 = i \mid M) \tag{E.2}$$

Propriété E.1.2 : contrainte
La définition E.1.1 d'une chaîne de Markov et ses paramètres définis en (E.1) et (E.2) implique que :

$$\forall i \in \{1, \ldots, N\}\,,\ \sum_{j=1}^{N} a_{M,ij} = 1 \text{ et } \sum_{j=1}^{N} \pi_{M,j} = 1$$

Par abus de notation, on écrira $a_{ij} = a_{M,ij}$, et $\pi_i = \pi_{M,i}$

La définition d'une chaîne de Markov simplifie l'écriture de la probabilité d'une séquence :

Propriété E.1.3 : probabilité d'une séquence
La définition E.1.1 d'une chaîne de Markov et ses paramètres définis en (E.1) et (E.2) implique que :

$$\mathbb{P}(s|M) = \mathbb{P}(q_1, \ldots, q_T \mid M) = \mathbb{P}(q_1 \mid M) \prod_{t=2}^{T} \mathbb{P}(q_t \mid \overline{q_{t-1}}, M) = \pi_{q_1} \prod_{t=2}^{T_s} a_{q_{t-1}, q_t}$$

E.1.2 Exemple : pièce de monnaie truquée

Enoncé

On considère une pièce truquée qui a 7 chances sur 10 de retomber sur pile (état 1), et 3 chances sur 10 de retomber sur face (état 2), et une vraie pièce. Si la face pile tombe, c'est la vraie pièce qui sera jouée, sinon, ce sera la pièce truquée. La première jouée est tirée au hasard. La chaîne de Markov correspondant à ce problème est définie par les probabilités suivantes :

$$\begin{array}{lll} \mathbb{P}(q_1 = 1) = 0,5 & \mathbb{P}(q_t = 1 \mid q_{t-1} = 1) = 0,5 & \mathbb{P}(q_t = 1 \mid q_{t-1} = 2) = 0,7 \\ \mathbb{P}(q_1 = 2) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 1) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 2) = 0,3 \end{array}$$

Par conséquent :

$$A = \begin{pmatrix} 0,5 & 0,5 \\ 0,7 & 0,3 \end{pmatrix} \quad \text{et} \quad \pi = \begin{pmatrix} 0,5 \\ 0,5 \end{pmatrix}$$

Ce modèle peut être représenté graphiquement par un schéma utilisant des graphes où chaque nœud est un état du modèle et chaque probabilité de transition positive un arc du graphe (voir figure E.1). Plus de détails sont donnés dans le paragraphe E.2.7.

Résultats intéressants

Cette modélisation simple permet d'obtenir la probabilité que la face pile apparaisse à un instant donné, et de définir le gain que peut espérer un tricheur en utilisant sa pièce truquée. On s'intéresse d'abord à :

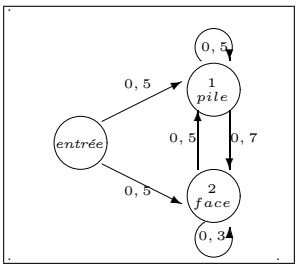


Fig. E.1: Représentation d'une chaîne de Markov sous forme de graphe.

$$\mathbb{P}\left(q_t = j \mid q_{t-2} = i, M\right) = \sum_{k=1}^{N} \mathbb{P}\left(q_t = j \mid q_{t-1} = k, M\right) \mathbb{P}\left(q_{t-1} = k \mid q_{t-2} = i, M\right) = \left(A_M^2\right)_{i,j}$$

Par récurrence, on en déduit que :

$$\mathbb{P}\left(q_t = j \mid q_{t-d} = i, M\right) = \left(A_M^d\right)_{i,j}$$

On note π'_M la transposée de la matrice π_M, on obtient alors que :

$$\pi'_M \left(A_M^t\right) = \left(\mathbb{P}\left(q_t = i \mid M\right)\right)_{1 \leqslant i \leqslant N}$$

Finalement, cette formule appliquée à l'exemple précédent donne :

$$\lim_{t \to +\infty} \mathbb{P}\left(q_t = 1\right) = \frac{7}{12} \text{ et } \lim_{t \to +\infty} \mathbb{P}\left(q_t = 2\right) = \frac{5}{12}$$

Les lois limites des états sont utiles pour calculer une espérence de gain. Si le joueur gagne un franc lorsque la face pile sort et perd un franc dans l'autre cas, sur une suite de douze coups, il gagne en moyenne deux francs. On peut également calculer la probabilité d'une séquence de quatre gain consécutifs :

$$\mathbb{P}\left(1111 \mid M\right) = 0,5 * 0,3 * 0,3 * 0,3 = 0,0135$$

E.2 Chaîne de Markov cachée

Il n'est pas évident qu'un processus (ou séquence de variables aléatoires) suive la loi d'une chaîne de Markov, néanmoins, ce processus peut parfois être expliqué par un autre caché qui suit la loi d'une chaîne de Markov. Ce second processus est dit caché car le premier, celui qu'il explique, est le seul observé. Afin de comprendre ce concept, l'exemple précédent (paragraphe E.1.2) sera légèrement modifié de manière à présenter les chaînes de Markov cachées.

E.2.1 Exemple : pièce de monnaie truquée

Dans l'exemple des deux pièces truquée et non truquée (paragraphe E.1.2), les états de la chaîne de Markov et faces des pièces étaient identiques. Les états correspondent maintenant aux pièces (information cachée) et les observations correspondent aux faces (information observée). L'état 1 sera la pièce non truquée et l'état 2 sera la pièce truquée. On suppose que la décision du joueur concernant le choix de la pièce ne

dépend plus de la face qui apparaît après le lancer mais dépend du choix de la pièce au lancer précédent. Les probabilités de transition sont définies ainsi :

$$\begin{array}{lll} \mathbb{P}(q_1 = 1) = 0,5 & \mathbb{P}(q_t = 1 \mid q_{t-1} = 1) = 0,5 & \mathbb{P}(q_t = 1 \mid q_{t-1} = 2) = 0,7 \\ \mathbb{P}(q_1 = 2) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 1) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 2) = 0,3 \end{array}$$

De chaque état dépendent les probabilités de voir apparaître pile ou face. On note O_t la face qui apparaît à l'instant t, les probabilités qui suivent sont appelées *probabilités d'émission.*

$$\begin{array}{ll} \mathbb{P}(O_t = 1 \mid q_t = 1) = 0,5 & \mathbb{P}(O_t = 1 \mid q_t = 2) = 0,7 \\ \mathbb{P}(O_t = 2 \mid q_t = 1) = 0,5 & \mathbb{P}(O_t = 2 \mid q_t = 2) = 0,3 \end{array}$$

Ce modèle peut toujours être représenté graphiquement par un schéma utilisant des graphes (figure E.2). Plus de détails seront donnés au paragraphe E.2.7.

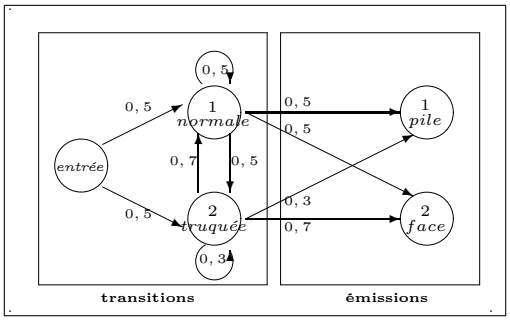


Fig. E.2: Représentation d'une chaîne de Markov sous forme de graphe.

On cherche alors à calculer probabilité de la séquence 1111 qui correspond à une série de quatre "face". Si la séquence d'observations est connue, il n'en est pas de même pour la séquence d'états (ou pièces) : il faut les envisager toutes et connaître la probabilité d'émettre une série de quatre "face" pour chacune d'elles. La réponse nécessite d'abord de définir ce qu'est mathématiquement une chaîne de Markov cachée car le calcul n'est plus aussi évident que pour celui des chaînes de Markov non cachées.

E.2.2 Définition d'une chaîne de Markov cachée

Une chaîne de Markov cachée part de l'idée que le processus stochatisque observé (la séquence d'observations) est expliqué par un autre processus (la séquence d'états) qui est inconnu.

Définition E.2.1 : chaîne de Markov cachée
Soit M une chaîne de Markov cachée,
Soit $Q = \{1, \cdots, N\}$ l'ensemble des états,
Soit $S = \overset{+\infty}{\underset{T=1}{\cup}} Q^T$ l'espace des séquences d'états,
Soit $\mathcal{O} = \{1, \cdots, D\}$ l'ensemble des observations,
Soit $\mathbf{O} = \overset{+\infty}{\underset{T=1}{\cup}} \mathcal{O}^T$ l'espace des séquences d'observations,
On note $s = (q_1, \ldots, q_{T_s}) \in S$ une séquence de longueur T_s,
Soit $O = (O_1, \ldots, O_{T_O}) \in \mathbf{O}$ une séquence de longueur T_O,
Alors une chaîne de Markov cachée est un modèle probabiliste vérifiant les quatre conditions suivantes :

1. L'observation à l'instant t ne dépend que de l'état à l'instant t :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall t \in \{1, \cdots, T_O\},\ \mathbb{P}\left(O_t | \overline{q_t}, \overline{O_{t-1}}, M\right) = \mathbb{P}\left(O_t | q_t, M\right)$$

   On appelle $\mathbb{P}\left(O_t | q_t, M\right)$ la *probabilité d'émission* de l'observation O_t sachant l'état q_t à l'instant t.

2. Les probabilités d'émissions ne dépendent pas du temps :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}\left(O_t | q_t, M\right) = \mathbb{P}\left(O_{t'} | q_{t'}, M\right)$$

3. L'état à l'instant t ne dépend que de l'état à l'instant $t-1$:

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall t \in \{2, \cdots, T_s\},\ \mathbb{P}\left(q_t | \overline{q_{t-1}}, \overline{O_{t-1}}, M\right) = \mathbb{P}\left(q_t | q_{t-1}, M\right)$$

   On appelle $\mathbb{P}\left(q_t | q_{t-1}, M\right)$ la *probabilité de transition* de l'état q_{t-1} à l'état q_t à l'instant t.

4. Les probabilités de transition ne dépendent pas du temps :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}\left(q_t | q_{t-1}, M\right) = \mathbb{P}\left(q_{t'} | q_{t'-1}, M\right)$$

Etant donné que ni les probabilités de transitions ni les probabilités d'émissions ne dépendent du temps, on définit pour le modèle M à N états, les matrices A_M, B_M et le vecteur Π_M par :

$$\begin{aligned}
A_M &= (a_{M,ij})_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant N}} = \left(\mathbb{P}\left(q_t = j \mid q_{t-1} = i, M\right)\right)_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant N}} \\
B_M &= (b_{M,ij})_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant D}} = \left(\mathbb{P}\left(O_t = j \mid q_t = i, M\right)\right)_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant j \leqslant D}} \\
\Pi_M &= (\pi_{M,i})_{1 \leqslant i \leqslant N} = \left(\mathbb{P}\left(q_1 = i \mid M\right)\right)_{1 \leqslant i \leqslant N}
\end{aligned}$$

La définition d'une chaîne de Markov cachée implique les contraintes suivantes sur les paramètres A_M,

B_M, Π_M résumées par la propriété suivante :

Propriété E.2.2 : contrainte
La définition E.2.1 et les notations définies en (E.3), (E.3) et (E.3) impliquent que :

$$\forall i \in \{1,\dots,N\}, \quad \sum_{j=1}^{N} a_{M,ij} = 1 \text{ et } \sum_{j=1}^{N} b_{M,ij} = 1$$
$$\sum_{i=1}^{N} \Pi_{M,i} = 1$$

Par abus de notation, lorsqu'il n'y a aucune ambiguïté, on note $A = A_M$, $B = B_M$, $\Pi = \Pi_M$. On cherche maintenant à exprimer la probabilité d'une séquence d'observations, soit $O \in \mathbf{O}$, on peut dorénavant définir :

$$\mathbb{P}(O \mid M) = \sum_{\substack{s\in S \\ T_s = T_O}} \mathbb{P}(O, s|M)$$
$$\mathbb{P}(O \mid M) = \sum_{\substack{s\in S \\ T_s = T_O}} \mathbb{P}(O_1,\dots,O_{T_O}, s_1,\dots,s_{T_s}|M)$$

En utilisant les hypothèses de la définition E.2.1, on cherche à exprimer cette probabilité à l'aide des paramètres A, B, Π du modèle M :

$$\mathbb{P}(O|M) = \sum_{\substack{s\in S \\ T_s = T_O}} \mathbb{P}(O, s|M)$$
$$\mathbb{P}(O|M) = \sum_{\substack{s\in S \\ T_s = T_O}} \mathbb{P}(O_{T_O}|s_{T_s}, M)\,\mathbb{P}(s_{T_s}|s_{T_s-1}, M)\,\mathbb{P}(O_1,\dots,O_{T_O-1}, s_1,\dots,s_{T_s-1}|M)$$
$$\mathbb{P}(O|M) = \sum_{\substack{s\in S \\ T_s = T_O}} \left[\mathbb{P}(s_1|M)\prod_{t=2}^{T_O}\mathbb{P}(s_t|s_{t-1}, M)\prod_{t=1}^{T_O}\mathbb{P}(O_t|s_t, M)\right]$$
$$\mathbb{P}(O|M) = \sum_{\substack{s\in S \\ T_s = T_O}} \left[\pi_{s_1}\prod_{t=2}^{T_O} a_{s_{t-1},s_t}\prod_{t=1}^{T_O} b_{q_t}(O_t)\right] \tag{E.3}$$

Néanmoins, cette expression (E.3) suppose un calcul coûteux en temps, il est nécessaire de factoriser certaines opérations.

E.2.3 Calcul factorisé de la probabilité d'une séquence

Soit $O = (O_1,\dots,O_T)$ une séquence d'observations, les séquences d'états seront notées $s = (q_1,\dots,q_T)$. On pose pour $1 \leqslant i \leqslant N$ et $1 \leqslant t \leqslant T$:

$$\alpha_t(i) = \alpha_{M,t}(i) = \mathbb{P}(q_t = i, O_1, ..., O_t \mid M) \tag{E.4}$$

Pour $t = 1$, on obtient pour tout $i \in \{1, \ldots, N\}$:

$$\begin{aligned} \alpha_1(i) &= \mathbb{P}(q_1 = i, O_1 \mid M) = \mathbb{P}(O_1 \mid q_1 = i, M)\,\mathbb{P}(q_1 = i \mid M) \\ &= \pi_i b_{i,O_1} \end{aligned} \tag{E.5}$$

On établit la récurrence suivante sur t et pour tout $j \in \{1, \ldots, N\}$:

$$\begin{aligned} \alpha_{t+1}(j) &= \mathbb{P}(q_{t+1} = j, O_1, ..., O_{t+1} \mid M) \\ \alpha_{t+1}(j) &= \mathbb{P}(O_{t+1} \mid q_{t+1} = j, O_1, ..., O_t, M)\,\mathbb{P}(q_{t+1} = j, O_1, ..., O_t \mid M) \\ \alpha_{t+1}(j) &= \mathbb{P}(O_{t+1} \mid q_{t+1} = j, M) \sum_{i=1}^{N} \mathbb{P}(q_{t+1} = j, q_t = i, O_1, ..., O_t \mid M) \\ \alpha_{t+1}(j) &= b_j(O_{t+1}) \sum_{i=1}^{N} \mathbb{P}(q_{t+1} = j \mid q_t = i, O_1, ..., O_t, M)\,\mathbb{P}(q_t = i, O_1, ..., O_t \mid M) \\ \alpha_{t+1}(j) &= b_j(O_{t+1}) \sum_{i=1}^{N} a_{ij}\,\alpha_t(i) \end{aligned} \tag{E.6}$$

Finalement, la probabilité de la séquence est obtenue grâce à la suite $\alpha_t(.)$ par un calcul appelé *forward* :

$$\mathbb{P}(O_1, ..., O_T \mid M) = \sum_{i=1}^{N} \alpha_T(i) \tag{E.7}$$

De ces formules, on tire l'algorithme suivant permettant de calculer la probabilité d'une séquence d'obser-

vations.

Algorithme E.2.3 : forward
Les notations utilisées sont celles des formules (E.4), (E.5), (E.6), (E.7).
Etape A : initialisation
pour $i = 1$ **à** N **faire**
$\alpha_1(i) \longleftarrow \pi_i b_{i,O_1}$
fin pour
Etape B : récurrence
pour $t = 2$ **à** T **faire**
pour $j = 1$ **à** N **faire**
$\alpha_t(j) \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$\alpha_t(j) \longleftarrow \alpha_t(j) + a_{ij}\,\alpha_{t-1}(i)$
fin pour
$\alpha_t(j) \longleftarrow \alpha_t(j)\ b_j(O_{t+1})$
fin pour
fin pour
Etape C : terminaison
$p \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$p \longleftarrow p + \alpha_T(i)$
fin pour
La probabilité de la séquence $(O_1, \ldots, O_T)$ est p obtenue à la dernière étape.

De la même manière, on définit pour $1 \leqslant i \leqslant N$ et $1 \leqslant t \leqslant T$ la suite :

$$\beta_t(i) = \beta_{M,t}(i) = \mathbb{P}(O_{t+1}, ..., O_T \mid q_t = i, M) \tag{E.8}$$

Par un calcul analogue à (E.5) et (E.6), on obtient pour tout $i \in \{1, \ldots, N\}$:

$$\beta_T(i) = \mathbb{P}(\emptyset \mid q_T = i, M) = 1 \tag{E.9}$$

$$\beta_t(i) = \sum_{j=1}^{N} b_j(O_{t+1})\ a_{ij}\,\beta_{t+1}(j) \tag{E.10}$$

Finalement, la probabilité de la séquence est également obtenue grâce à la suite $\beta_t(.)$ par un calcul appelé *backward* :

$$\mathbb{P}(O_1, ..., O_T \mid M) = \sum_{i=1}^{N} \pi_i\,\beta_1(i)\ b_i(O_1) \tag{E.11}$$

Les fonctions $\alpha_t(.)$ et $\beta_t(.)$ permettent de calculer la probabilité d'une séquence avec un coût en $O\left(TN^2\right)$ opérations. Ces calculs sont semblables à des algorithmes de programmation dynamique et parfois appelés algorithmes *forward* ($\alpha_t(.)$) et *backward* ($\beta_t(.)$) ([Rabiner1986]). De ces formules, on tire l'algorithme

suivant permettant de calculer la probabilité d'une séquence d'observations.

Algorithme E.2.4 : backward
Les notations utilisées sont celles des formules (E.8), (E.9), (E.10), (E.11).
Etape A : initialisation
pour $i = 1$ **à** N **faire**
$\beta_T(i) \longleftarrow 1$
fin pour
Etape B : récurrence
pour $t = T-1$ **à** 1 **faire**
pour $i = 1$ **à** N **faire**
$\beta_t(j) \longleftarrow 0$
pour $j = 1$ **à** N **faire**
$\beta_t(i) \longleftarrow \beta_t(i) + a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$
fin pour
fin pour
fin pour
Etape C : terminaison
$p \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$p \longleftarrow p + \beta_1(i)\, b_i(O_1)\, \pi_i$
fin pour
La probabilité de la séquence $(O_1, \ldots, O_T)$ est p obtenue à la dernière étape.

E.2.4 Autres résultats intéressants

Trois autres résultats intéressants utilisés lors de l'apprentissage (paragraphe E.4.1) peuvent être obtenus de manière similaire :

$$\mathbb{P}(O_1, ..., O_T, q_t = i \mid M) = \alpha_t(i)\, \beta_t(i) \tag{E.12}$$

$$\forall t \in \{1, \ldots, T\},\ \mathbb{P}(O_1, ..., O_T \mid M) = \sum_{i=1}^{N} \alpha_t(i)\, \beta_t(i) \tag{E.13}$$

$$\mathbb{P}(O_1, ..., O_T, q_t = i, q_{t+1} = j \mid M) = \alpha_t(i)\ a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j) \tag{E.14}$$

E.2.5 Retour à l'exemple

Rappel des probabilités de transitions et d'émission :

$$\begin{array}{lll} \mathbb{P}(q_1 = 1) = 0,5 & \mathbb{P}(q_t = 1 \mid q_{t-1} = 1) = 0,5 & (q_t = 1 \mid q_{t-1} = 2) = 0,7 \\ \mathbb{P}(q_1 = 2) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 1) = 0,5 & \mathbb{P}(q_t = 2 \mid q_{t-1} = 2) = 0,3 \end{array}$$

$$\begin{array}{ll} \mathbb{P}(O_t = 1 \mid q_t = 1) = 0,7 & \mathbb{P}(O_t = 1 \mid q_t = 2) = 0,5 \\ \mathbb{P}(O_t = 2 \mid q_t = 1) = 0,3 & \mathbb{P}(O_t = 2 \mid q_t = 2) = 0,5 \end{array}$$

On cherche à calculer la probabilitée de la séquence 1111 :

	$\alpha_1(.)$	$\alpha_2(.)$	$\alpha_3(.)$	$\alpha_4(.)$
état 1	$0,15$	$0,094$	$0,04099$	$0,0174454$
état 2	$0,25$	$0,085$	$0,03535$	$0,014986$

$$P(1111) = 0,0174454 + 0,014986 = 0,0324314$$

Les lois limites des observations peuvent également être obtenus :

$$\begin{aligned}
\mathbb{P}(O=k|t) &= \mathbb{P}(O=k|q=1,t)\,\mathbb{P}(q=1|t) + \mathbb{P}(O=k|q=2,t)\,\mathbb{P}(q=2|t) \\
\lim_{t\to\infty}\mathbb{P}(O=k|t) &= \lim_{t\to\infty}\left[\mathbb{P}(O=k|q=1,t)\,\mathbb{P}(q=1|t) + \mathbb{P}(O=k|q=2,t)\,\mathbb{P}(q=2|t)\right] \\
\lim_{t\to\infty}\mathbb{P}(O=k|t) &= \mathbb{P}(O=k|q=1,t)\lim_{t\to\infty}\mathbb{P}(q=1|t) + \mathbb{P}(O=k|q=2,t)\lim_{t\to\infty}\mathbb{P}(q=2|t)
\end{aligned}$$

Comme :

$$\begin{aligned}
\lim_{t\to+\infty} P(q_t=1) &= \tfrac{7}{12} \\
\lim_{t\to+\infty} P(q_t=2) &= \tfrac{5}{12}
\end{aligned}$$

On en déduit que :

$$\begin{aligned}
\lim_{t\to+\infty} P(O_t=1) &= 0,7 * \tfrac{7}{12} + 0,5 * \tfrac{5}{12} = \tfrac{37}{60} \\
\lim_{t\to+\infty} P(O_t=2) &= 0,3 * \tfrac{7}{12} + 0,5 * \tfrac{5}{12} = \tfrac{23}{60}
\end{aligned}$$

E.2.6 Introduction d'un état d'entrée et d'un état de sortie

En reconnaissance de l'écriture, les séquences d'observations ne dépassent pas quelques graphèmes par lettres : toutes les séquences d'observations sont finies, or cette information supplémentaire n'est pas prise en compte dans les chaînes de Markov cachées présentées jusqu'à présent. L'introduction d'un état d'entrée et d'un état de sortie va y remédier afin de signifier la fin de la séquence (voir [Chen1994]). La figure E.3 montre les deux modèles optimaux (avec ou sans état de sortie) pour la lettre "M". Le dessin de cette lettre fait intervenir trois graphèmes identiques. Le premier modèle (1) sans état de sortie ne peut prendre en compte la "durée" de la lettre "M", des séquences de deux, trois, cent graphèmes auront toutes la même probabilité. Le second modèle (2) ne permet qu'une seule écriture de la lettre "M" en trois graphèmes. Tous les états de la chaîne de Markov sont des états *émetteurs* (voir définition E.2.5) car chaque observation est associé un état, les états d'entrées et de sortie sont *non émetteurs* (voir définition E.2.6).

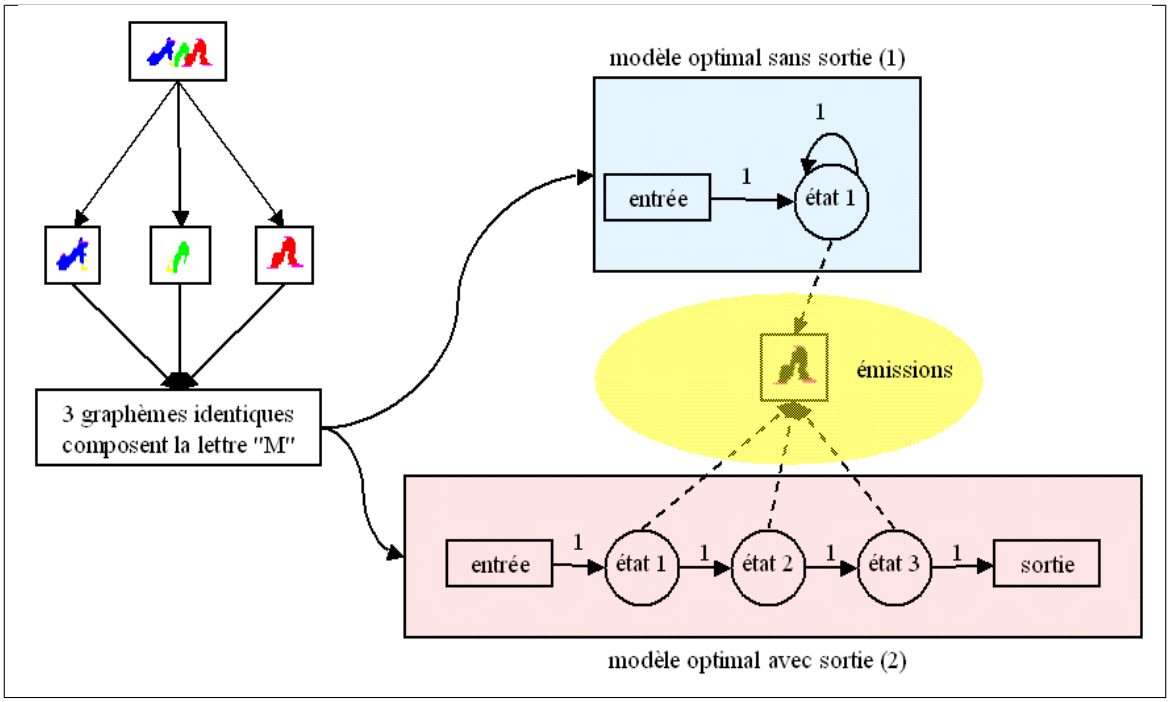


Fig. E.3: Modèles optimaux pour la lettre "M" avec et sans état de sortie

Définition E.2.5 : état émetteur
Un état d'une chaîne de Markov cachée est dit *émetteur* si le passage par cet état implique l'émission d'une observation. Par définition, pour une séquence d'observations O de longueur T, toutes les séquences d'états cachés permises pour cette séquence O contiennent exactement T états émetteurs.

Définition E.2.6 : état non émetteur
Un état d'une chaîne de Markov cachée est dit *non émetteur* (ou *muet*) si le passage par cet état n'implique aucune émission d'observation. Par définition, pour toute séquence d'observations, une séquence d'états cachés peut contenir une infinité d'états non émetteurs.

Définition E.2.7 : chaîne de Markov cachée, entrée et sortie (ES)
Soit M une chaîne de Markov cachée (ES),
Soit $Q = \{1, \cdots, N\}$ l'ensemble des états,
Soit $S = \overset{+\infty}{\underset{T=1}{\cup}} Q^T$ l'espace des séquences d'états,
Soit $\mathcal{O} = \{1, \cdots, D\}$ l'ensemble des observations,
Soit $\mathbf{O} = \overset{+\infty}{\underset{T=1}{\cup}} \mathcal{O}^T$ l'espace des séquences d'observations,
On note $s = (q_1, \ldots, q_{T_s}) \in S$ une séquence de longueur T_s,
Soit $O = (O_1, \ldots, O_{T_O}) \in \mathbf{O}$ une séquence de longueur T_O,
Alors une chaîne de Markov cachée est un modèle probabiliste vérifiant les quatre conditions suivantes :

1. L'observation à l'instant t ne dépend que de l'état à l'instant t :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall t \in \{1, \cdots, T_O\},\ \mathbb{P}\left(O_t | \overline{q_t}, \overline{O_{t-1}}, M\right) = \mathbb{P}\left(O_t | q_t, M\right)$$

   On appelle $\mathbb{P}\left(O_t | q_t, M\right)$ la *probabilité d'émission* de l'observation O_t sachant l'état q_t à l'instant t.

2. Les probabilités d'émissions ne dépendent pas du temps :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}\left(O_t | q_t, M\right) = \mathbb{P}\left(O_{t'} | q_{t'}, M\right)$$

3. L'état à l'instant t ou la sortie ne dépend que de l'état à l'instant $t-1$:

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall t \in \{2, \cdots, T_s\},\quad \mathbb{P}\left(q_t | \overline{q_{t-1}}, \overline{O_{t-1}}, M\right) = \mathbb{P}\left(q_t | q_{t-1}, M\right)$$
$$\forall s \in S \text{ telle que } T_s = T_O, \forall t \in \{2, \cdots, T_s\},\quad \mathbb{P}\left(sortie | \overline{q_{t-1}}, \overline{O_{t-1}}, M\right) = \mathbb{P}\left(sortie | q_{t-1}, M\right)$$

   On appelle $\mathbb{P}\left(q_t | q_{t-1}, M\right)$ la probabilité de transition de l'état q_{t-1} à l'état q_t à l'instant t et $\mathbb{P}\left(sortie | q_{t-1}, M\right) = \mathbb{P}\left(s | q_{t-1}, M\right)$ la *probabilité de sortie* à l'instant $t-1$.

4. Les probabilités de transition et de sortie ne dépendent pas du temps :

$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}\left(q_t | q_{t-1}, M\right) = \mathbb{P}\left(q_{t'} | q_{t'-1}, M\right)$$
$$\forall s \in S \text{ telle que } T_s = T_O,\ \forall \left(t, t'\right) \in \{2, \cdots, T_s\},\ \mathbb{P}\left(sortie | q_{t-1}, M\right) = \mathbb{P}\left(sortie | q_{t'-1}, M\right)$$

La chaîne de Markov cachée (ES) M à N états est définie par les paramètres A_M, B_M, Π_M, Θ_M :

$$A_M = (a_{M,ij})_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}} = (\mathbb{P}(q_t = j \mid q_{t-1} = i, M))_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}} \tag{E.15}$$

$$B_M = (b_{M,ij})_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant D}} = (\mathbb{P}(O_t = j \mid q_t = i, M))_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant D}} \tag{E.16}$$

$$\Pi_M = (\pi_{M,i})_{1\leqslant i\leqslant N} = (\mathbb{P}(q_1 = i \mid M))_{1\leqslant i\leqslant N} \tag{E.17}$$

$$\Theta_M = (\theta_{M,i})_{1\leqslant i\leqslant N} = (\mathbb{P}(s \mid q_t, M))_{1\leqslant i\leqslant N} \tag{E.18}$$

La définition d'une chaîne de Markov cachée (ES) implique les contraintes suivantes sur les paramètres A_M, B_M, Π_M, Θ_M résumées par la propriété suivante :

Propriété E.2.8 : contrainte
La défintion E.2.7 et les notations définies en (E.15), (E.16), (E.17) et (E.18) impliquent que :

$$\forall i \in \{1,\ldots,N\}, \quad \sum_{j=1}^{N} a_{M,ij} + \theta_{M,i} = 1 \tag{E.19}$$

$$\forall i \in \{1,\ldots,N\}, \quad \sum_{j=1}^{N} b_{M,ij} = 1 \tag{E.20}$$

$$\sum_{i=1}^{N} \Pi_{M,i} = 1 \tag{E.21}$$

En utilisant les hypothèses de la définition E.2.1, on cherche à exprimer la probabilité d'une séquence d'observations à l'aide des paramètres $A = A_M$, $B = B_M$, $\Pi = \pi_M$, $\Theta = \Theta_M$ du modèle M :

$$\begin{aligned}\mathbb{P}(O|M) &= \sum_{\substack{s\in S\\T_s=T_O}} \mathbb{P}(O,s|M)\\ \mathbb{P}(O|M) &= \sum_{\substack{s\in S\\T_s=T_O}} \left[\pi_{s_1}\theta_{s_T}\prod_{t=2}^{T_O} a_{s_{t-1},s_t}\prod_{t=1}^{T_O} b_{q_t}(O_t)\right]\end{aligned} \tag{E.22}$$

Le calcul factorisé de cette probabilité est aussi modifié, les suites $\alpha_t(.)$ et $\beta_t(.)$ deviennent :

Calcul de la suite $\alpha_t(.)$
Pour $1 \leqslant i \leqslant N$ et $1 \leqslant t \leqslant T$,

$$\alpha_t(i) = \mathbb{P}(q_t = i, O_1, ..., O_t \mid M) \tag{E.23}$$

L'initialisation ne change pas :

$$\alpha_1(i) = \mathbb{P}(q_1 = i, O_1 \mid M) = \mathbb{P}(O_1 \mid q_1 = i, M)\,\mathbb{P}(q_1 = i \mid M) = \pi_i b_i(O_1) \tag{E.24}$$

Par récurrence :

$$\alpha_{t+1}(j) = b_{j,O_{t+1}} \sum_{i=1}^{N} a_{ij} \alpha_t(i) \tag{E.25}$$

Seule change la probabilité de la séquence :

$$\mathbb{P}(O_1, ..., O_T \mid M) = \sum_{i=1}^{N} \alpha_T(i)\ \theta_i \tag{E.26}$$

L'algorithme E.2.3 devient le suivant :

Algorithme E.2.9 : forward
Les notations utilisées sont celles des formules (E.23), (E.24), (E.25), (E.26).
Etape A : initialisation
pour $i = 1$ **à** N **faire**
$\alpha_1(i) \longleftarrow \pi_i b_{i,O_1}$
fin pour
Etape B : récurrence
pour $t = 2$ **à** T **faire**
pour $j = 1$ **à** N **faire**
$\alpha_t(j) \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$\alpha_t(j) \longleftarrow \alpha_t(j) + a_{ij}\, \alpha_{t-1}(i)$
fin pour
$\alpha_t(j) \longleftarrow \alpha_t(j)\ b_j(O_{t+1})$
fin pour
fin pour
Etape C : terminaison
$p \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$p \longleftarrow p + \alpha_T(i)\ \theta_i$
fin pour
La probabilité de la séquence $(O_1, \ldots, O_T)$ est p obtenue à la dernière étape.

Calcul de la suite $\beta_t(.)$

Pour $1 \leqslant i \leqslant N$ et $1 \leqslant t \leqslant T$,

$$\beta_t(i) = \mathbb{P}(O_{t+1}, ..., O_T \mid q_t = i, M) \tag{E.27}$$

L'initialisation change :

$$\begin{aligned} &\beta_T(i) = \mathbb{P}(\emptyset \mid q_T = i, M) = \theta_i \\ &\qquad (= \text{probabilité que la séquence soit finie sachant que } q_T = i) \end{aligned} \tag{E.28}$$

Par récurrence :

$$\beta_t(i) = \sum_{j=1}^{N} b_j(O_{t+1})\ a_{ij}\ \beta_{t+1}(j) \tag{E.29}$$

Finalement :

$$\mathbb{P}(O_1, ..., O_T \mid M) = \sum_{i=1}^{N} \pi_i\, \beta_1(i)\ b_{i,O_1} \tag{E.30}$$

L'algorithme E.2.4 devient le suivant :

Algorithme E.2.10 : backward
Les notations utilisées sont celles des formules (E.27), (E.28), (E.29), (E.30).
Etape A : initialisation
pour $i = 1$ **à** N **faire**
$\beta_T(i) \longleftarrow \theta_i$
fin pour
Etape B : récurrence
pour $t = T - 1$ **à** 1 **faire**
pour $i = 1$ **à** N **faire**
$\beta_t(j) \longleftarrow 0$
pour $j = 1$ **à** N **faire**
$\beta_t(i) \longleftarrow \beta_t(i) + a_{ij}\, b_j(O_{t+1})\ \beta_{t+1}(j)$
fin pour
fin pour
fin pour
Etape C : terminaison
$p \longleftarrow 0$
pour $i = 1$ **à** N **faire**
$p \longleftarrow p + \beta_1(i)\ b_i(O_1)\ \pi_i$
fin pour
La probabilité de la séquence $(O_1, \ldots, O_T)$ est p obtenue à la dernière étape.

Par la suite, toutes les chaînes de Markov seront supposées posséder un état d'entrée et un état de sortie.

E.2.7 Représentation d'une chaîne de Markov sous forme de graphe

Les paragraphes précédents ont déjà montré qu'il était possible de modéliser une chaîne de Markov sous forme de graphe où les noeuds sont les états et les transitions les arcs. Une probabilité non nulle de passer d'un état i à un autre état j peut être envisagée comme un lien unidirectionnel entre ces deux états dont le poids est la probabilité de transition de l'état i vers l'état j. L'ensemble des probabilités non nulles d'un modèle définit un ensemble de liens entre les états qui peut être décrit par un graphe. Un modèle entièrement connecté de N états contient $N^2 + 2N$ connexions. Une structure de graphe permet de diminuer ce nombre de connexions en ne tenant compte que des connexions non nulles (les modèles utilisés pour la reconnaissance de l'écriture contiennent en général une grande part de connexions nulles.), voir figures E.4, E.5.

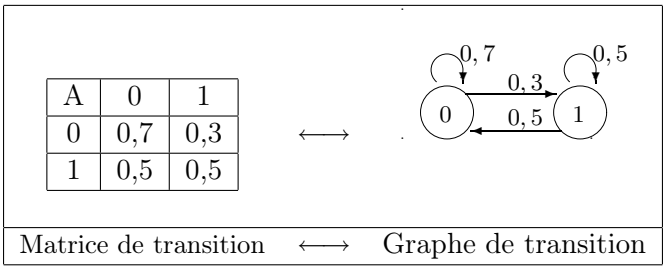


Fig. E.4: Equivalence entre matrice de transition et graphe de transition pour une chaîne de Markov.

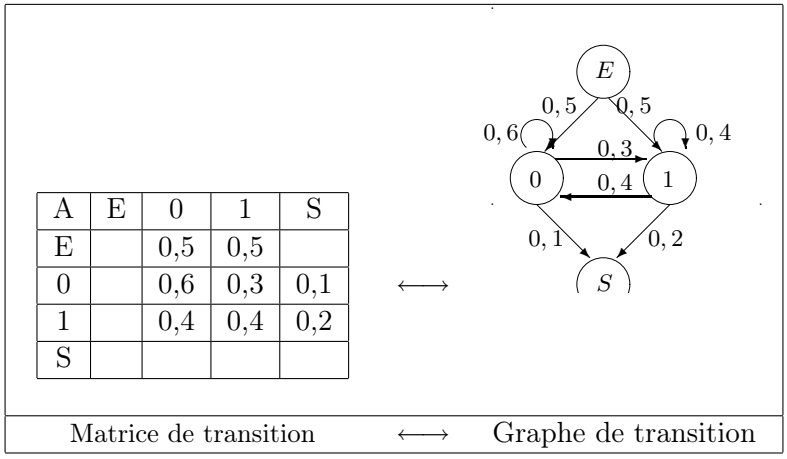


Fig. E.5: Equivalence entre matrice de transition et graphe de transition pour une chaîne de Markov ES.

La reconnaissance de l'écriture utilise peu de modèles ergodiques (ou entièrement connectés) car le sens de la lecture interdit de revenir à un état déjà visité. Par conséquent, les matrices de connexions sont triangulaires supérieures avec des zéros sur la diagonale. En définitive, il existe peu de connexions non nulles par rapport à toutes celles qui sont possibles. Le tableau E.1 (page 261) montre que, en général, seules 10% des connexions possibles sont non nulles : la description sous forme de graphe de ces chaînes de Markov cachée est plus avantageuse qu'une description matricielle. Ce résultat est bien sûr propre à la reconnaissance de l'écriture manuscrite.

E.3 Algorithme du meilleur chemin : algorithme de Viterbi

Nous avons vu que le calcul des suites $\alpha_t(.)$ et $\beta_t(.)$ permet de calculer la probabilité d'une séquence d'observations, qui est une somme de probabilités sur l'ensemble des séquences d'états possibles. L'algorithme de Viterbi permet de trouver parmi toutes ces séquences d'états, celle dont la probabilité d'émettre la séquence d'observations est la plus forte. On appelle aussi cette séquence d'états ou meilleur chemin la séquence d'états la plus probable ayant émis la séquence d'observations. Soit une séquence d'observations $O = (O_1, ..., O_T)$, et le modèle M, cet algorithme permet de trouver la séquence $s^*(O_1, ..., O_T, M)$:

$$s^*(O_1, ..., O_T, M) = \arg\max_s \mathbb{P}(s \mid O_1, ..., O_T, M) = \arg\max_s \mathbb{P}(s, O_1, ..., O_T \mid M) \tag{E.31}$$

On note $(\delta_t(i))_{\substack{1 \leqslant t \leqslant T \\ 1 \leqslant i \leqslant N}}$ la probabilité de la séquence d'états $(q_1, ..., q_t)$ la plus probable telle que $q_t = i$ ayant émis la séquence $(O_1, ..., O_t)$:

lettre	**nombre d'états**	**connexions possibles**	**connexions non nulles**	**rapport**
A	42	1848	150	$8,1\%$
B	36	1368	112	$8,2\%$
C	27	783	76	$9,7\%$
D	30	960	97	$10,1\%$
E	23	575	73	$12,7\%$
F	34	1224	96	$7,8\%$
G	38	1520	126	$8,3\%$
H	36	1368	97	$7,1\%$
I	24	624	70	$11,2\%$
J	10	120	21	$17,5\%$
K	32	1088	80	$7,4\%$
L	30	960	36	$3,8\%$
M	42	1848	126	$6,8\%$
N	36	1368	103	$7,5\%$
O	27	783	71	$9,1\%$
P	35	1295	100	$7,7\%$
Q	34	1224	85	$6,9\%$
R	33	1155	108	$9,4\%$
S	29	899	92	$10,2\%$
T	31	1023	80	$7,8\%$
U	27	783	62	$7,9\%$
V	29	899	77	$8,6\%$
W	36	1368	78	$5,7\%$
X	36	1368	97	$7,1\%$
Y	40	1680	114	$6,8\%$
Z	38	1520	90	$5,9\%$

Tab. E.1: Connexions non nulles dans les modèles de reconnaissance de lettres.

$$\delta_t(i) = \underset{(q_1,\ldots,q_{t-1})}{\arg\max}\ \mathbb{P}(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=i|M) \tag{E.32}$$

alors $\delta_t(i)$ vérifie :

$$\begin{array}{rl} \text{pour } t=1 \text{ et } 1\leqslant i\leqslant N, & \delta_1(i)=\pi_1 b_i(O_1) \\ \text{pour } 2\leqslant t\leqslant T \text{ et } 1\leqslant j\leqslant N, & \delta_t(j)=\max\limits_{1\leqslant i\leqslant N}\{\delta_{t-1}(i)\,a_{ij}\,b_i(O_t)\} \end{array} \tag{E.33}$$

On définit également la suite $(\lambda_t)_{1\leqslant t\leqslant T}$ par :

$$\begin{array}{rl} \text{pour } 1\leqslant t\leqslant T-1, & \lambda_t=\underset{1\leqslant i\leqslant N}{\arg\max}\{\delta_t(\lambda_{t+1})\} \\ \text{pour } t=T, & \lambda_T=\underset{1\leqslant i\leqslant N}{\arg\max}\{\delta_T(i)\,\theta_i\} \end{array} \tag{E.34}$$

Par conséquent, le meilleur chemin est la séquence d'états $(\lambda_1,...,\lambda_T)$ et a pour probabilité $\delta_T(\lambda_T)\,\theta_{\lambda_T}$.

On en déduit l'algorithme suivant :

Algorithme E.3.1 : Viterbi
Les notations utilisées sont celles des équations (E.31), (E.32), (E.33), (E.34).

Etape A : initialisation
pour $i = 1$ **à** N **faire**
$\delta_1(i) \longleftarrow \pi_i\, b_i(O_1)$
$\lambda_1(i) \longleftarrow -1$
fin pour

Etape B : récurrence
pour $t = 2$ **à** T **faire**
pour $j = 1$ **à** N **faire**
$\delta_t(j) \longleftarrow \delta_{t-1}(1)\ a_{1j}\, b_j(O_t)$
$\lambda_t(j) \longleftarrow 1$
pour $i = 2$ **à** N **faire**
$x \longleftarrow \delta_{t-1}(i)\ a_{ij}\, b_j(O_t)$
si $x < \delta_t(j)$ **alors**
$\delta_t(j) \longleftarrow x$
$\lambda_t(j) \longleftarrow i$
fin si
fin pour
fin pour
fin pour

Etape C : terminaison
$\delta_{T+1} \longleftarrow \delta_T(1)\ \theta_1$
$\lambda_{T+1} \longleftarrow 1$
pour $i = 2$ **à** N **faire**
$x \longleftarrow \delta_T(i)\ \theta_i$
si $x < \delta_{T+1}$ **alors**
$\delta_{T+1} \longleftarrow x$
$\lambda_{T+1} \longleftarrow i$
fin si
fin pour

Etape D : séquence d'états la plus probable
$q_T^* \longleftarrow \lambda_{T+1}$
pour $t = T - 1$ **à** 1 **faire**
$q_t^* \longleftarrow \lambda_{t+1}\left(q_{t+1}^*\right)$
fin pour

La séquence d'états la plus probable est $(q_1^*, \ldots, q_T^*)$ et a pour probabilité δ_{T+1}.

Remarque E.3.2: forward et backward
L'obtention du meilleur chemin nécessite deux passages, le premier pour le calcul des matrices $\delta_t(i)$ et $\lambda_t(i)$ lors des étapes A, B, C. Ce calcul est semblable à celui de l'algorithme forward E.2.9. Le second passage de l'étape D dans l'autre sens (indice décroissant) permet de retrouver le meilleur chemin. Cette étape n'est pas nécessaire pour obtenir seulement la probabilité de la meilleur séquence d'états.

Remarque E.3.3: meilleure séquence, plus court chemin
Cet algorithme est à rapprocher d'un algorithme de recherche du plus court chemin dans un graphe. En effet, la probabilité d'un chemin s'exprime comme un produit de probabilités :

$$\mathbb{P}(q_1, ..., q_T, O_1, ..., O_T \mid M) = \pi_{q_1} b_{q_1}(O_1) \prod_{t=2}^{T} a_{q_{t-1}, q_t} b_{q_t}(O_t)$$

En passant au logarithme, on obtient une somme de termes qui peuvent être considérés comme des distances entre deux états. L'algorithme de Viterbi n'est autre qu'un algorithme de recherche du meilleur chemin de type Dijkstra ([Dijkstra1971]).

E.4 Apprentissage d'une chaîne de Markov cachée

E.4.1 Principe

Dans l'exemple paragraphe E.2.1, la chaîne de Markov cachée adaptée au problème se déduisait de l'énoncé : les probabilités de transitions et d'émissions étaient fixées par la définition des deux pièces truquée et non truquée. Les questions que l'on cherche à résoudre dans ces problèmes sont en général des espérances de gain, des durées, des informations sur le comportement du jeu sur une longue période, sur de longues séquences d'observations. A partir du modèle, on cherche donc à déduire des propriétés sur les observations.

L'apprentissage d'une chaîne de Markov cachée est exactement la tâche inverse. On dispose de séquences d'observations dont il faut déduire le modèle qui les a générées. Une fois la topologie du modèle choisie, l'apprentissage revient donc à estimer les probabilités d'entrées, de transitions, d'émissions qui modélisent au mieux la base d'échantillons.

Fig. E.6: Un mot segmenté en graphèmes.

La figure E.6 est un exemple de mot à reconnaître, la reconnaissance avec dictionnaire (deux mots pour cet exemple) consiste à reconnaître que c'est le mot "CHARLES" plutôt que "JEROME" qui est écrit sur cette image.

Pour répondre à cette question, deux modèles de Markov cachés sont construits, l'un pour le mot "CHARLES", $M_{CHARLES}$, et l'autre pour le mot "JEROME", M_{JEROME}. Le prétraitement de l'image aboutit à la séquence d'observations $O = (O_1, \ldots, O_T)$. On dit que :

$$\text{si } \mathbb{P}\left(O|M_{CHARLES}\right) > \mathbb{P}\left(O|M_{JEROME}\right) \text{ alors l'image contient le mot "CHARLES"}$$

Il reste à construire les modèles $M_{CHARLES}$ et M_{JEROME} et pour cela on dispose d'une base d'images annotées qui contiennent des images des mots "CHARLES" et "JEROME". Le mot $M_{CHARLES}$ va apprendre toutes les séquences issues des images annotées "CHARLES", il en sera de même pour le mot "JEROME". Si on note $\left(O_1^C, \ldots, O_K^C\right)$ les séquences d'observations annotées "CHARLES" et $\left(O_1^J, \ldots, O_L^J\right)$ celles annotées "JEROME", l'apprentissage consiste à maximiser la vraisemblance :

$$\begin{aligned} L\left(\theta_C, \theta_J, O_1^C, \ldots, O_K^C, O_1^J, \ldots, O_K^J\right) &= \prod_{n=1}^{K} \mathbb{P}\left(O_n^C|M_{CHARLES}\right) \prod_{n=1}^{L} \mathbb{P}\left(O_n^L|M_{JEROME}\right) \\ &= \prod_{n=1}^{K} \mathbb{P}\left(O_n^C|\theta_C\right) \prod_{n=1}^{L} \mathbb{P}\left(O_n^L|\theta_J\right) \\ &= L\left(\theta_C, O_1^C, \ldots, O_K^C\right) L\left(\theta_J, O_1^J, \ldots, O_K^J\right) \\ &\quad \text{où } \theta_C \text{ et } \theta_J \text{ sont les paramètres} \\ &\quad \text{des modèles } M_{CHARLES} \text{ et } M_{JEROME} \end{aligned} \tag{E.35}$$

L'apprentissage soulève deux questions :

1. Le choix des modèles pour les mots "CHARLES" et "JEROME", ce point sera étudié dans la partie F.
2. L'apprentissage de ces modèles, ce point est détaillé dans les paragraphes qui suivent (algorithme, convergence, démonstration).

L'équation (E.35) suggère que l'apprentissage des modèles "CHARLES" et "JEROME" peut s'effectuer de manière indépendante à condition que les vraisemblances associées à ces deux modèles dépendent de paramètres différents. Dans le cas contraire, la résolution du problème se déduit du cas où on suppose qu'un seul modèle M doit apprendre les séquences d'observations :

$$\left(O^k = \left(O_1^k, ..., O_{T_k}^k\right)\right)_{1 \leqslant k \leqslant K}$$

Problème E.4.1 : apprentissage d'une chaîne de Markov cachée
Les notations utilisées sont celles de la définition E.2.1, l'équation (E.35) permet de définir l'apprentissage d'une chaîne de Markov cachée comme étant la solution du problème d'optimisation suivant :

$$(A_M, \pi_M, \theta_M, B_M) = \underset{A,\pi,\theta,B}{\arg\max} \underbrace{\prod_{k=1}^{K} \mathbb{P}\left(O_1^k, ..., O_{T_k}^k \,|M\right)}_{\text{vraisemblance du modèle}}$$

$$\text{avec les contraintes} \quad \begin{cases} \sum\limits_i \pi_i = 1 \\ \forall j, \sum\limits_i a_{ij} = 1 \\ \forall j, \sum\limits_o b_j(o) = 1 \\ \forall i, \pi_i \geqslant 0 \\ \forall i, \theta_i \geqslant 0 \\ \forall (i,j)\ a_{ij} \geqslant 0 \\ \forall (i,o)\ b_j(o) \geqslant 0 \end{cases}$$

L'algorithme d'optimisation ou apprentissage des modèles de Markov cachés est basé sur les formules de Baum-Welch qui prennent en compte les contraintes du problème (voir [Baum1972] ou [Rabiner1986]), utilisées comme un cas particulier de l'algorithme EM (Expectation-Maximisation, voir [Dempster1977]).

$$\overline{a_{i,j}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k - 1} \alpha_t^k(i)\ a_{i,j}\, b_j\left(O_{t+1}^k\right)\ \beta_{t+1}^k(j) \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i)\, \beta_t^k(i) \right]} \qquad \overline{b_i(o)} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i)\, \beta_t^k(i)\ \mathbb{1}_{\{O_t^k = o\}} \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i)\, \beta_t^k(i) \right]}$$

$$\overline{\theta_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \alpha_{T_k}^k(i)\ \overbrace{\beta_{T_k}^k(i)}^{=\theta_i}}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i)\, \beta_t^k(i) \right]} \qquad \overline{\pi_i} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{P_k} \overbrace{\alpha_1^k(i)}^{=\pi_i} \beta_1^k(i)$$

avec $P_k = P\left(O_1^k, \ldots, O_{T_k}^k\right)$ et $\alpha_t^k(i) = \mathbb{P}\left(O_1^k, \ldots, O_t^k, q_t = i | M\right)$

$\beta_t^k(i) = \mathbb{P}\left(O_{t+1}^k, \ldots, O_{T_k}^k | q_t = i, M\right)$

Tab. E.2: Formules de réestimation de Baum-Welch.

Trois étapes composent cet algorithme itératif :

Algorithme E.4.2 : apprentissage d'une chaîne de Markov cachée
Cet algorithme permet d'obtenir une solution au problème E.4.1 correspondant à un minimum local de la vraisemblance (E.35) comme le montre le théorème E.4.3) :

Etape A : initialisation

Les paramètres $A_0, \Pi_0, \Theta_0, B_0$ reçoivent des valeurs aléatoires.

$t \longleftarrow 0$

calcul de la vraisemblance du modèle L_0

Etape B : récurrence

tant que $(L_t > L_{t-1})$ **faire**

$t \longleftarrow t + 1$

Les paramètres $\overline{A_t}, \overline{\Pi_t}, \overline{\Theta_t}, \overline{B_t}$ sont estimés en fonction des formules de Baum-Welch (voir table E.2).

$A_t \longleftarrow \overline{A_t}$

$\Pi_t \longleftarrow \overline{\Pi_t}$

$\Theta_t \longleftarrow \overline{\Theta_t}$

$B_t \longleftarrow \overline{B_t}$

calcul de la vraisemblance du modèle L_t

fin tant que

Théorème E.4.3 : convergence de l'algorithme E.4.2
Soit $M = (A, B, \Theta, \Pi)$ une chaîne de Markov et $\left(O^k = \left(O_1^k, ..., O_{T_k}^k\right)\right)_{1 \leqslant k \leqslant K}$ une suite de séquences d'observations, l'algorithme E.4.2 implique la convergence croissante de la vraisemblance :

$$\prod_{k=1}^{K} \mathbb{P}\left(O_1^k, ..., O_{T_k}^k \,|M\right) \tag{E.36}$$

Remarque E.4.4: convergence vers un minimum local
Le théorème E.4.3 démontre de la suite $(L_t)_{t \leqslant 0}$ construite par l'algorithme E.4.2, la valeur atteinte correspond à un minimum local et non global de la vraisemblance (E.36).

Les paragraphes qui suivent (E.4...) donnent différentes démonstrations de ce théorème.

E.4.2 Démonstration intuitive

Démonstration (théorème E.4.3) :

L'étape B de l'algorithme d'apprentissage E.4.2 consiste à réestimer les paramètres A, π, θ, B de manière à accroître la vraisemblance $L(A, \pi, \theta, B)$:

$$L(A, \pi, \theta, B) = \prod_{k=1}^{K} \mathbb{P}\left(O_1^k, ..., O_{T_k}^k \,|M\right)$$

$$L(A, \pi, \theta, B) = \prod_{k=1}^{K} \left[\sum_{s \in S_p} rO_1^k, ..., O_{T_k}^k, s \,|M\right]$$

où s est l'ensemble des séquences d'états de la chaîne de Markov cachée M

Le principe des formules de Baum-Welch consiste à augmenter la valeur des paramètres très probables et à diminuer celle de ceux peu probables. On note :

- $l_{i,t}$ le nombre de chemins (ou séquences) partant de l'état i à l'instant t (voir figure E.7)
- $l_{i,j,t}$ le nombre de chemins partant de l'état i à l'instant t et passant à l'état j à l'instant $t+1$ (voir figure E.7)

La nouvelle valeur $\overline{a_{ij}}$ sera : $\overline{a_{ij}} = \dfrac{\sum\limits_t l_{i,j,t}}{\sum\limits_t l_{i,t}}$. Il reste à exprimer cette expression en termes de probabilités :

$$\overline{a_{i,j}} = \frac{\sum\limits_{k=1}^{K} \frac{1}{P_k} \sum\limits_{t=1}^{T_k - 1} \mathbb{P}\left(q_{t+1} = j, q_t = i, O^k\right)}{\sum\limits_{k=1}^{K} \frac{1}{P_k} \sum\limits_{t=1}^{T_k} \mathbb{P}\left(q_t = i, O^k\right)} = \frac{\sum\limits_{k=1}^{K} \frac{1}{P_k} \left[\sum\limits_{t=1}^{T_k - 1} \alpha_t^k(i)\ a_{i,j}\, b_{j,O_{t+1}}\, \beta_{t+1}^k(j)\right]}{\sum\limits_{k=1}^{K} \frac{1}{P_k} \left[\sum\limits_{t=1}^{T_k} \alpha_t^k(i)\, \beta_t^k(i)\right]}$$

d'après l'expression (E.14), page 253

La réestimation des probabilités d'émission suit le même raisonnement, pour alléger les notations, on note $O = O^k$ et $C_o^t = \mathbb{1}_{\{O_t = o\}}$:

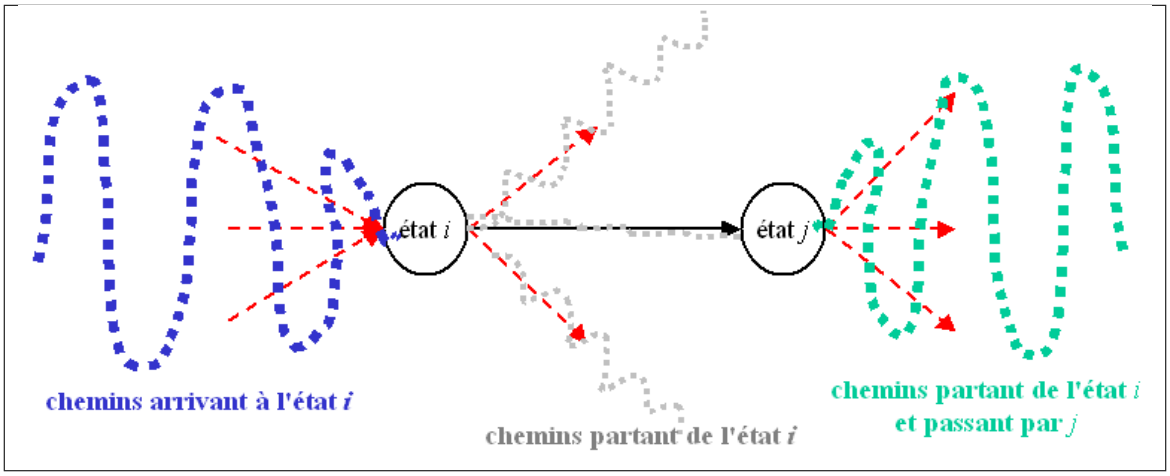


Fig. E.7: Idée des formules de Baum-Welch : donner une nouvelle valeur à un coefficient tenant compte du nombre de chemins qui l'empruntent.

$$\begin{aligned}
\mathbb{P}\left(O_t = o, q_t, O\right) &= \mathbb{P}\left(C_o^t, q_t, O\right) = \mathbb{P}\left(O_{t+1}, .., O_T \left| C_o^t, q_t, O_1, ..., O_t\right.\right) \mathbb{P}\left(C_o^t, q_t, O_1, ..., O_t\right) \\
\mathbb{P}\left(C_o^t, q_t, O\right) &= \mathbb{P}\left(O_1, \ldots, O_T | q_t\right) \mathbb{P}\left(C_o^t | q_t, O_1, \ldots, O_t\right) \mathbb{P}\left(q_t, O_1, \ldots, O_t\right) \\
\mathbb{P}\left(C_o^t, q_t, O\right) &= \beta_t\left(q_t\right) \mathbb{1}_{\{O_t = o\}} \alpha_t\left(q_t\right)
\end{aligned}$$

D'où :

$$\overline{b_i\left(o\right)} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[P\left(O_t = o, q_t = i, O\right)\right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} P\left(q_t = i, O\right)\right]} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k\left(i\right) \beta_t^k\left(i\right) \mathbb{1}_{\{O_t^k = o\}}}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k\left(i\right) \beta_t^k\left(i\right)}$$

On peut calculer de même :

$$\overline{\theta_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \mathbb{P}\left(q_{T_k} = i, O\right)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \mathbb{P}\left(q_t = i, O^k\right)} \text{ et } \overline{\pi_i} = \sum_{k=1}^{K} \frac{1}{P_k} \mathbb{P}\left(q_1 = i, O^k\right)$$

(E.4.3) □

E.4.3 Lemmes et théorèmes intermédiaires

Ces lemmes servent des démonstrations plus rigoureuses exposées au paragraphe suivant (E.4.4).

Lemme E.4.5 : Levinson1983 (1)
(voir [Levinson1983])
Soit $(u_1, ..., u_N) \in \left(\mathbb{R}_+^*\right)^N$ et $(v_1, ..., v_N) \in (\mathbb{R}_+)^N$ tels que $\sum_{i=1}^{N} v_i > 0$ alors :

$$\ln \left[\frac{\sum_{i=1}^{N} v_i}{\sum_{i=1}^{N} u_i} \right] \geqslant \frac{\sum_{i=1}^{N} u_i \ln v_i - u_i \ln u_i}{\sum_{i=1}^{N} u_i}$$

Démonstration (lemme E.4.5) :
On utilise la concavité de la fonction logarithme :

$$\ln \left[\frac{\sum_{i=1}^{N} v_i}{\sum_{i=1}^{N} u_i} \right] = \ln \left[\overbrace{\sum_{i=1}^{N} \frac{u_i}{\sum_{k=1}^{N} u_k} \frac{v_i}{u_i}}^{\text{moyenne pondérée}} \right] \geqslant \sum_{i=1}^{N} \frac{u_i}{\sum_{k=1}^{N} u_k} \ln \frac{v_i}{u_i} = \frac{1}{\sum_{k=1}^{N} u_k} \sum_{i=1}^{N} u_i \ln v_i - u_i \ln u_i$$

(E.4.5) □

Lemme E.4.6 : Levinson1983 (2)
(voir [Levinson1983]) Soit $(u_1, ..., u_N) \in \left(\mathbb{R}_+^*\right)^N$, la solution unique de la maximisation sous contrainte suivante :

$$\left| \begin{array}{l} \max\limits_{(x_1,...,x_N)} \sum\limits_{i=1}^{N} u_i \ln x_i \\ \sum\limits_{i=1}^{N} x_i = 1 \end{array} \right.$$

est obtenue pour $\forall i \in \{1, ..., N\}$, $x_i = \dfrac{u_i}{\sum_{i=1}^{N} u_i}$

Démonstration (lemme E.4.6) :

On utilise les multiplicateurs de Lagrange, on pose : $F(x_1, ..., x_N, \lambda) = \sum_{i=1}^{N} u_i \ln x_i + \lambda \left(\sum_{i=1}^{N} x_i - 1 \right)$

Lorsque F est maximum, ses dérivées partielles vérifient :

$$\frac{\partial F(x_1, ..., x_N, \lambda)}{\partial x_k} = \frac{u_k}{x_k} + \lambda = 0 \iff x_k = -\frac{u_k}{\lambda}$$

$$\implies -\sum_{k=1}^{N} \frac{u_k}{\lambda} = -1 \implies \lambda = -\sum_{k=1}^{N} u_k \implies x_k = \frac{u_k}{\sum_{k=1}^{N} u_k}$$

(E.4.6) □

Lemme E.4.7 : multiplicateurs de Lagrange
La solution du problème de maximisation suivant :

$$\left| \begin{array}{l} \max\limits_{(x_1,...,x_N)} f\left(x_1,...,x_N\right) \\ \sum\limits_{i=1}^{N} x_i = 1 \end{array} \right.$$

vérifie $\sum\limits_{k=1}^{N} x_k \dfrac{\partial f}{\partial x_k}\left(x_1,...,x_N\right) \neq 0 \Longrightarrow \forall i \in \{1,\cdots,N\},\ x_i = \dfrac{x_i \dfrac{\partial f}{\partial x_i}\left(x_1,...,x_N\right)}{\sum\limits_{k=1}^{N} x_k \dfrac{\partial f}{\partial x_k}\left(x_1,...,x_N\right)}$

Démonstration (lemme E.4.7) :

On utilise les multiplicateurs de Lagrange, on pose : $F\left(x_1,...,x_N,\lambda\right) = f\left(x_1,...,x_N\right) + \lambda\left(\sum\limits_{i=1}^{N} x_i - 1\right)$
Lorsque F est maximum, ses dérivées partielles vérifient (on pose $X = (x_1,\ldots,x_N)$:

$$\begin{aligned} \frac{\partial F\left(X,\lambda\right)}{\partial x_k} = \frac{\partial f}{\partial x_k}\left(X\right) + \lambda = 0 &\Longrightarrow x_k \frac{\partial f}{\partial x_k}\left(X\right) + \lambda x_k = 0 \\ &\Longrightarrow \sum_{k=1}^{N}\left[x_k \frac{\partial f}{\partial x_k}\left(X\right) + \lambda x_k\right] = 0 \\ &\Longrightarrow \sum_{k=1}^{N} x_k \frac{\partial f}{\partial x_k}\left(X\right) = -\lambda \end{aligned}$$

En remplaçant λ par $-\sum\limits_{k=1}^{N} x_k \dfrac{\partial f}{\partial x_k}(...)$ dans chacune des équations aux dérivées partielles, on démontre le lemme E.4.7. Cette optimisation revient à chercher le maximum de la fonction f sur l'hyperplan d'équation $\sum\limits_{i=1}^{N} x_i = 1$.

(E.4.7) □

C'est ce lemme qui est à la source du théorème E.4.10 mais au préalable suivent une définition et un

lemme.

Définition E.4.8 : fonction log-log-convexe
Soit une fonction :

$$\begin{array}{rccc} f: & \left(\mathbb{R}_+^*\right)^n & \longrightarrow & \mathbb{R}+^* \\ & (x_1,\ldots,x_n) & \longrightarrow & f\left(x_1,\ldots,x_n\right) \end{array}$$

f est une fonction *log-log-convexe* si et seulement si la fonction :

$$\begin{array}{rccc} g: & (\mathbb{R})^n & \longrightarrow & \mathbb{R}+^* \\ & (u_1,\ldots,u_n) & \longrightarrow & g\left(u_1,\ldots,u_n\right) = \ln\left[f\left(e^{u_1},\ldots,e^{u_n}\right)\right] \end{array}$$

est une fonction convexe.

Lemme E.4.9 : distance de Kullback-Leiber
On pose $D = \left\{ x = (x_1,\ldots,x_n) \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} x_i = 1 \text{ et } \forall i \in \{1,\cdots,n\}, x_i > 0 \right\}$.
Soit $(x,y) \in D^2$ alors :

$$\sum_{i=1}^{n} y_i \ln\left[\frac{x_i}{y_i}\right] \leqslant 0$$

Démonstration (lemme E.4.9) :
La fonction $x \longrightarrow \ln x$ est concave, donc :

$$\begin{aligned} \sum_{i=1}^{n} y_i \ln\left[\frac{x_i}{y_i}\right] &= \sum_{i=1}^{n} \frac{y_i}{\sum_{k=0}^{n} y_k} \ln\left[\frac{x_i}{y_i}\right] \text{ car } y \in D \\ &\leqslant \ln\left[\sum_{i=1}^{n} \frac{y_i}{\sum_{k=0}^{n} y_k} \frac{x_i}{y_i}\right] \\ &\leqslant \ln\left[\frac{\sum_{i=1}^{n} x_i}{\sum_{i=0}^{n} y_i}\right] = \ln\frac{1}{1} \text{ car } (x,y) \in D^2 \\ &\leqslant 0 \end{aligned}$$

(E.4.9) □

Théorème E.4.10 : fonction log-log-convexe
On pose $D = \left\{ x = (x_1, \ldots, x_n) \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} x_i = 1 \text{ et } \forall i \in \{1, \cdots, n\}, x_i > 0 \right\}$.
Soit $f : \left(\mathbb{R}_+^*\right)^n \longrightarrow \mathbb{R}+^*$ une fonction log-log-convexe dérivable.
Soit $x \in D$, on définit $y(x) = (y_1(x), \ldots, y_n(x)) \in D$ tel que :

$$\forall i \in \{1, \cdots, n\},\ y_i(x) = \frac{x_i \frac{\partial f}{\partial x_i}(x)}{\sum_{k=1}^{n} x_k \frac{\partial f}{\partial x_k}(x)}$$

alors f vérifie :

$$\forall x \in D,\ f(y(x)) \geqslant f(x)$$

Démonstration (théorème E.4.10) :

Soit $(x, x') \in D^2$, on définit $(u, u') \in (\mathbb{R}^n)^2$ tel que :

$$\forall i \in \{1, \cdots, n\},\ u_i = \ln x_i \text{ et } u'_i = \ln x'_i$$

u et u' vérifient :

$$\sum_{i=1}^{n} e^{u_i} = 1 \text{ et } \sum_{i=1}^{n} e^{u'_i} = 1$$

On pose $h(u) = \ln f(e^{u_1}, \ldots, e^{u_n})$, f est log-log-convexe, donc :

$$\begin{aligned}
h\left(u'\right) - h(u) &\geqslant \sum_{i=1}^{n} \frac{\partial h}{\partial u_i}(u)\left(u'_i - u_i\right) \\
\ln f\left(x'\right) - \ln f(x) &\geqslant \sum_{i=1}^{n} \frac{e^{u_i}}{f(x)} \frac{\partial f}{\partial x_i}(x)\left(\ln x'_i - \ln x_i\right) \\
\ln\left[\frac{f(x')}{f(x)}\right] &\geqslant \sum_{i=1}^{n} \frac{x_i}{f(x)} \frac{\partial f}{\partial x_i}(x) \ln\left[\frac{x'_i}{x_i}\right]
\end{aligned}$$

Si $x' = y(x) = \left(y_i(x) = \frac{x_i \frac{\partial f}{\partial x_i}(x)}{\sum_{k=1}^{n} x_k \frac{\partial f}{\partial x_k}(x)} \right)_{1 \leqslant i \leqslant n}$, alors :

$$
\begin{array}{rrcl}
 & -\ln\left[\dfrac{f\left(y\left(x\right)\right)}{f\left(x\right)}\right] & \leqslant & \displaystyle\sum_{i=1}^{n}\frac{x_i}{f\left(x\right)}\frac{\partial f}{\partial x_i}\left(x\right)\ln\left[\frac{x_i}{y_i\left(x\right)}\right] \\
\Longrightarrow & -\ln\left[\dfrac{f\left(y\left(x\right)\right)}{f\left(x\right)}\right] & \leqslant & \left[\displaystyle\sum_{i=1}^{n}\frac{x_i\frac{\partial f}{\partial x_i}\left(x\right)}{f\left(x\right)}\right]\underbrace{\left[\displaystyle\sum_{i=1}^{n}y_i\left(x\right)\ln\left[\frac{x_i}{y_i\left(x\right)}\right]\right]}_{\leqslant 0 \text{ d'après le lemme E.4.9}} \\
\Longrightarrow & \ln\left[\dfrac{f\left(y\left(x\right)\right)}{f\left(x\right)}\right] & \geqslant & 0 \\
\Longrightarrow & f\left(y\left(x\right)\right) & \geqslant & f\left(x\right)
\end{array}
$$

(E.4.10) □

Corollaire E.4.11 : polynôme à coefficients positifs (Baum1968)
(voir [Baum1968])
Soit $P:\mathbb{R}^N\longrightarrow\mathbb{R}$ un polynôme dont les coefficients sont tous positifs, soit $x=(x_1,\ldots,x_N)\in\mathbb{R}^N$ tels que $\sum_{i=1}^{N}x_i=1$ et $\forall i\in\{1,\cdots,N\},\ x_i\geqslant 0$, soit $y=(y_1,\ldots,y_N)\in\mathbb{R}^N$ défini par :

$$\forall i\in\{1,\cdots,N\},\ y_i=\frac{x_i\frac{\partial P}{\partial x_i}\left(x\right)}{\sum_{k=1}^{N}x_k\frac{\partial P}{\partial x_k}\left(x\right)}$$

alors :

$$P\left(y\right)\geqslant P\left(x\right)$$

Démonstration (corollaire E.4.11) :

Soit D l'ensemble défini dans le théorème E.4.10, si f est une fonction log-log-convexe défini sur $\overline{D}$, si f est continue alors les égalités démontrées sur D le sont aussi sur $\overline{D}$. Il ne reste plus qu'à démontrer qu'un polynôme à coefficients positifs est log-log-convexe.

Soit $x=(x_1,\ldots,x_n)\in D$, on note $e^u=(e_{u_1},\ldots,e_{u_n})$, on peut écrire le polynôme P sous la forme :

$$
\begin{aligned}
P\left(x\right) &= \sum_{0\leqslant i_1,\ldots,i_n\leqslant\deg P}a_{i_1,\ldots,i_n}\prod_{k=1}^{n}x_k^{i_k} \\
P\left(e^u\right) &= \sum_{0\leqslant i_1,\ldots,i_n\leqslant\deg P}a_{i_1,\ldots,i_n}\prod_{k=1}^{n}\left(e^{u_k}\right)^{i_k} \\
P\left(e^u\right) &= \sum_{0\leqslant i_1,\ldots,i_n\leqslant\deg P}a_{i_1,\ldots,i_n}e^{\sum_{k=1}^{n}i_ku_k} \\
\frac{\partial P\left(e^u\right)}{\partial u_l} &= \sum_{0\leqslant i_1,\ldots,i_n\leqslant\deg P}a_{i_1,\ldots,i_n}\,i_l\,e^{\sum_{k=1}^{n}i_ku_k}
\end{aligned}
$$

On pose $i=(i_1,\ldots,i_n)$. On en déduit pour $m\in\{1,\cdots,n\}$ que :

$$\frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} = \frac{1}{P^2\left(e_u\right)} \left[\begin{array}{l} \left(\sum_i a_i\, i_l i_m\, e^{\langle i,u\rangle}\right)\left(\sum_i a_i\, e^{\langle i,u\rangle}\right) \\ - \left(\sum_i a_i\, i_l\, e^{\langle i,u\rangle}\right)\left(\sum_i a_i\, i_m\, e^{\langle i,u\rangle}\right) \end{array} \right]$$

$$\frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} = \frac{1}{P^2\left(e_u\right)} \left[\sum_i \sum_j a_i\, a_j\, i_l\left(i_m - j_m\right)\, e^{\langle i,u\rangle}\, e^{\langle j,u\rangle} \right]$$

Il faut montrer que la matrice $\left(\frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m}\right)_{u_l u_m}$ est définie positive. On cherche donc à montrer que pour tout $y \in \mathbb{R}^n$, $\sum_{l,m} y_l y_m \frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} \geqslant 0$. On note $b_{i_1,\ldots,i_n} = a_{i_1,\ldots,i_n} e^{\sum_{k=1}^{n} i_k u_k} \geqslant 0$.

$$\begin{aligned}
\sum_{l,m} y_l y_m \frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} &= \sum_{l,m} y_l y_m \sum_i \sum_j a_i\, a_j\, i_l\left(i_m - j_m\right)\, e^{\langle i,u\rangle}\, e^{\langle j,u\rangle} \\
&= \sum_i \sum_j \sum_{l,m} y_l y_m\, b_i b_j i_l\left(i_m - j_m\right) \\
&= \sum_i \sum_j b_i b_j \left(\sum_{l,m} y_l y_m i_l\left(i_m - j_m\right) \right) \\
&= \sum_i \sum_j b_i b_j \left(\left[\sum_l y_l i_l\right]\left[\sum_m y_m i_m\right] - \left[\sum_l y_l i_l\right]\left[\sum_m y_m j_m\right] \right)
\end{aligned}$$

On pose $I_i = \sum_l y_l i_l$, comme $\frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} = \frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_m \partial u_l}$, on peut écrire que :

$$\begin{aligned}
\sum_{l,m} y_l y_m \frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} &= \frac{1}{2} \sum_{l,m} y_l y_m \left[\frac{\partial\left(\ln P\left(e^{u}\right)\right)}{\partial u_l \partial u_m} + \frac{\partial\left(\ln \mathbb{P}\left(e^{u}\right)\right)}{\partial u_m \partial u_l} \right] \\
&= \frac{1}{2} \sum_i \sum_j b_i b_j \left(I_i^2 - I_i I_j + I_j^2 - I_j I_i \right) \\
&= \frac{1}{2} \sum_i \sum_j b_i b_j \left(I_i - I_j \right)^2 \\
&\geqslant 0
\end{aligned}$$

Un polynôme à coefficients positifs est donc log-log-convexe. **(E.4.11)** □

E.4.4 Démonstration basée sur le gradient

Cette démonstration est présentée dans [Levinson1983].

Démonstration (théorème E.4.3) :

L'expression de la probabilité d'une séquence O connaissant le modèle M est :

$$\mathbb{P}\left(O\left|M\right.\right) = \sum_{(q_1,...,q_T)} \left[\pi_{q_1} b_{q_1}\left(O_1\right) \left(\prod_{t=1}^{T-1} a_{q_t,q_{t+1}} b_{q_{t+1}}\left(O_t\right) \right) \theta_{q_T} \right]$$

Chaque coefficient intervient plusieurs fois dans l'expression de la probabilité, cependant il ne peut intervenir qu'une seule fois à chaque temps $u \in \{1, ...T\}$. C'est pourquoi on décompose $P\left(O\left|M\right.\right)$ en :

$$\mathbb{P}\left(O\left|M\right.\right) = \sum_{u=1}^{T-1} \left[\sum_{(i,j)} a_{ij} b_j\left(O_t\right) \underbrace{\sum_{(q_1,...,q_T)}^{q_u=i,\, q_{u+1}=j} \pi_{q_1} b_{q_1}\left(O_1\right) \theta_{q_T} \left(\prod_{t=1, t\neq u}^{T-1} a_{q_t,q_{t+1}} b_{q_{t+1}}\left(O_t\right) \right)}_{=\mathbb{P}(q_u=i,\, q_{u+1}=j, O\left|M\right.)} \right]$$

$$\iff \mathbb{P}\left(O\left|M\right.\right) = \sum_{u=1}^{T-1} \left[\sum_{(i,j)} a_{ij} b_j\left(O_t\right) \alpha_u\left(i\right) \beta_{u+1}\left(j\right) \right] \tag{E.37}$$

$$\iff \mathbb{P}\left(O\left|M\right.\right) = \sum_{u=1}^{T} \sum_{i,o} b_i\left(o\right) \underbrace{\sum_{(q_1,...,q_T)}^{q_u=i,\, O_u=o} \left[\pi_{q_1} b_{q_1}\left(O_1\right) \left(\prod_{t=1}^{T-1} a_{q_t,q_{t+1}} b_{q_{t+1}}\left(O_t\right) \right) \theta_{q_T} \right]}_{=\mathbb{P}(q_u=i,\, O_u=o, O\left|M\right.)=\mathbb{P}(q_u=i, O\left|M\right.)\, \mathbb{1}_{\{O_u=o\}}} \tag{E.38}$$

On en déduit que :

$$\begin{aligned}
(E.37) \Longrightarrow \quad \frac{\partial \mathbb{P}\left(O\left|M\right.\right)}{\partial a_{ij}} &= \sum_{u=1}^{T-1} b_j\left(O_t\right) \alpha_u\left(i\right) \beta_{u+1}\left(j\right) \\
(E.37) \Longrightarrow \quad \frac{\partial \mathbb{P}\left(O\left|M\right.\right)}{\partial \pi_i} &= \beta_1\left(i\right) \\
(E.37) \Longrightarrow \quad \frac{\partial \mathbb{P}\left(O\left|M\right.\right)}{\partial \theta_i} &= \alpha_T\left(i\right) b_i\left(O_T\right) \\
(E.38) \Longrightarrow \quad \frac{\partial \mathbb{P}\left(O\left|M\right.\right)}{\partial b_i\left(o\right)} &= \sum_{u=1}^{T} \alpha_u\left(i\right) \beta_u\left(i\right) \mathbf{1}_{\{O_u=o\}}
\end{aligned}$$

De plus, dans le cas de plusieurs séquences, si x est un coefficient du modèle, on utilise le fait que :

$$\frac{\partial L}{\partial x} = \frac{\partial \left(\prod_{k=1}^{K} \mathbb{P}\left(O_1^k, ..., O_{T_k}^k \left|M\right.\right) \right)}{\partial x} = \sum_{k=1}^{K} \frac{L}{\mathbb{P}\left(O^k\left|M\right.\right)} \frac{\partial \mathbb{P}\left(O^k\left|M\right.\right)}{\partial x}$$

Comme la fonction $(A, B, \theta, \pi) \longrightarrow \mathbb{P}\left(O|M\right)$ est un polynôme à coefficients positifs, le théorème E.4.11 nous assure de la croissance de $\mathbb{P}\left(O|M\right)$ au cours des itérations de l'algorithme d'apprentissage. Comme cette suite est majorée par 1, elle est convergente.

(E.4.3) □

E.4.5 Démonstration antérieure à la découverte de l'algorithme EM

Cette démonstration est présentée dans [Levinson1983].

Démonstration (théorème E.4.3) :

Soient deux modèles M et M' possédant le même nombre d'états. On définit $P_k(M) = \mathbb{P}\left(O^k | M\right)$ et $P_k(M') = \mathbb{P}\left(O^k | M'\right)$. Pour cette séquence d'observations, il existe N^{T_k} séquences d'états possibles. On note $s^i = \left(s_1^i, ..., s_{T_k}^i\right)$ la séquence d'état d'indice i.

$$P_k(M) = \sum_{(q_1,...,q_{T_k})} \left[\pi_{q_1} b_{q_1}\left(O_1^k\right) \left(\prod_{t=1}^{T_k-1} a_{q_t,q_{t+1}} b_{q_{t+1}}\left(O_t^k\right) \right) \theta_{q_{T_k}} \right] = \sum_{i=1}^{N^{T_k}} u_i^k$$

De manière analogue :

$$P_k\left(M'\right) = \sum_{(q_1,...,q_T)} \left[\pi'_{q_1} b'_{q_1}\left(O_1^k\right) \left(\prod_{t=1}^{T_k-1} a'_{q_t,q_{t+1}} b'_{q_{t+1}}\left(O_t^k\right) \right) \theta'_{q_{T_k}} \right] = \sum_{i=1}^{N^{T_k}} v_i^k$$

En appliquant le lemme E.4.5, on trouve :

$$\ln\left(\frac{\prod_{k=1}^{K} P_k(M')}{\prod_{k=1}^{K} P_k(M)} \right) = \sum_{k=1}^{K} \ln\left(\frac{\sum_{i=1}^{N^{T_k}} v_i^k}{\sum_{i=1}^{N^{T_k}} u_i^k} \right) \geqslant \sum_{k=1}^{K} \frac{\sum_{i=1}^{N^{T_k}} u_i^k \ln v_i^k - u_i^k \ln u_i^k}{\sum_{i=1}^{N^{T_k}} u_i^k} = \sum_{k=1}^{K} \left[\frac{Q_k(M,M')}{P_k(M)} - \frac{Q_k(M,M)}{P_k(M)} \right]$$

On cherche à maximiser :

$$\max_{(v_1,...,v_{N^T})} \sum_{k=1}^{K} \frac{Q_k(M,M')}{P_k(M)} = \max_{(v_1,...,v_{N^T})} \sum_{k=1}^{K} \left[\frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \ln v_i^k \right]$$

Or :

$$u_i^k \ln v_i^k = u_i^k \left[\ln \pi'_{q_{s_1^i}} + \ln b'_{s_1^i}\left(O_1^k\right) + \sum_{t=1}^{T_k-1} \left(\ln a'_{s_t^i, s_{t+1}^i} + \ln b'_{s_{t+1}^i}\left(O_t^k\right) \right) + \ln \theta'_{s_T^i} \right]$$

On cherche à écrire différemment la somme $\sum_{i=1}^{N^{T_k}} u_i^k \ln v_i^k$ sous la forme :

$$\sum_{k=1}^{K} \left[\frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \ln v_i^k \right] = \sum_{n=1}^{N} C_n \ln \pi'_n + \sum_{n=1}^{N}\sum_{m=1}^{N} A_{nm} \ln a'_{nm} + \sum_{n=1}^{N}\sum_{o} B_{n,o} \ln b'_n(o) + \sum_{n=1}^{N} D_n \ln \theta'_n \quad \text{(E.39)}$$

Pour cela, on note :

- $p(s,i,j)$ avec $(i,j) \in \{E,S,1,...,N\}^2$ le nombre de fois où la connexion $i \to j$ est utilisée dans la séquence d'états s
- $p'(s,i,o)$ avec $(i,o) \in \{E,S,1,...,N\} \times \mathbf{O}$ le nombre de fois où l'état i émet l'observation o dans la séquence d'états s

Avec ces notations :

$$A_{nm} = \sum_{k=1}^{K} \frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \, p\left(s^i, n, m\right) = \sum_{k=1}^{K} \frac{1}{P_k(M)} \mathbb{P}\left(q_t = n, q_{t+1} = m, O^k \,|M\right) \tag{E.40}$$

$$B_n = \sum_{k=1}^{K} \frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \, p'\left(s^i, n, o\right) = \sum_{k=1}^{K} \frac{1}{P_k(M)} \mathbb{P}\left(q_t = n, O_t = o, O^k \,|M\right) \tag{E.41}$$

$$C_n = \sum_{k=1}^{K} \frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \, p\left(s^i, E, k\right) = \sum_{k=1}^{K} \frac{1}{P_k(M)} \mathbb{P}\left(q_1 = n, O^k \,|M\right) \tag{E.42}$$

$$D_n = \sum_{k=1}^{K} \frac{1}{P_k(M)} \sum_{i=1}^{N^{T_k}} u_i^k \, p\left(s^i, n, S\right) = \sum_{k=1}^{K} \frac{1}{P_k(M)} \mathbb{P}\left(q_{T_k} = n, O^k \,|M\right) \tag{E.43}$$

Les équations (E.12) et (E.14) (page 253) permettent de déduire les valeurs de A_{nm}, B_n, C_n, D_n. En appliquant le lemme E.4.6 à l'équation (E.39), on démontre que les valeurs qui maximisent $Q(M, M')$ sont :

$$a'_{ij} = \frac{A_{ij}}{\sum_{l=1}^{N} A_{il}} \qquad b'_{i,o} = \frac{B_{i,o}}{\sum_{r \in \mathbf{O}} B_{i,r}}$$

$$\pi'_i = \frac{C_i}{\sum_{l=1}^{N} C_l} \qquad \theta'_i = \frac{D_i}{\sum_{l=1}^{N} D_l}$$

On retrouve les formules de Baum-Welch. Si on note (M_t) la suite de modèles obtenus après chaque itération, la démonstration précédente (paragraphe E.4.4) nous assure que la suite $\mathbb{P}(O \,|M_t)$ est croissante, comme elle est majorée par un, elle est convergente. Cependant, la convergence s'effectue vers un maximum local.

(E.4.3) □

E.4.6 Algorithme EM (Expectation-Maximisation)

E.4.6.1 Définition

Dans le cas général, l'algorithme permet d'estimer les paramètres d'une loi lorsque certaines données sont manquantes ou cachées. On considère deux variables aléatoires :

- $X \in \mathcal{X} \subset \mathbb{R}^p$ de densité $f(x \,|\theta)$ avec $\theta \in \Theta$ (Θ est l'ensemble des paramètres)
- $Z \in \mathcal{Z} \subset \mathbb{R}^q$ de densité $g(z \,|\theta)$

Les variables X, Z, (X, Z) sont appelées :

- X est la variable observée ou incomplète
- Z est la variable cachée ou manquante
- (X, Z) est la variable complète

On note $h(x, z \,|\theta)$ la densité de (X, Z) et $k(z \,|x, \theta)$ la densité de la variable $E(Z \,|X)$. D'après la règle de Bayes :

$$
\begin{aligned}
& h(x,z\,|\theta) = k(z\,|x,\theta)\, f(x\,|\theta) \\
\Longrightarrow\ & f(x\,|\theta) = \frac{h(x,z\,|\theta)}{k(z\,|x,\theta)} \\
\Longrightarrow\ & \ln f(x\,|\theta) = \ln h(x,z\,|\theta) - \ln k(z\,|x,\theta) \\
\Longrightarrow\ & [\ln f(x\,|\theta)]\, k(z\,|x,\theta) = [\ln h(x,z\,|\theta)]\, k(z\,|x,\theta) - [\ln k(z\,|x,\theta)]\, k(z\,|x,\theta) \\
\Longrightarrow\ & \int_{\mathcal{Z}} [\ln f(x\,|\theta)]\, k(z\,|x,\theta)\, dz = \int_{\mathcal{Z}} [\ln h(x,z\,|\theta)]\ k(z\,|x,\theta)\, dz - \int_{\mathcal{Z}} [\ln k(z\,|x,\theta)]\ k(z\,|x,\theta)\, dz
\end{aligned}
$$

On définit :

$$
\begin{aligned}
Q_{\mathcal{Z}}(\varphi,\theta) &= E_{\mathcal{Z}}[\ln h(x,z\,|\varphi)\ |\, x,\theta] = \int_{\mathcal{Z}} [\ln h(x,z\,|\varphi)]\ k(z\,|x,\theta)\, dz \\
H_{\mathcal{Z}}(\varphi,\theta) &= E_{\mathcal{Z}}[\ln h(z\,|x,\varphi)\ |\, x,\theta] = \int_{\mathcal{Z}} [\ln k(z\,|x,\varphi)]\ k(z\,|x,\theta)\, dz
\end{aligned}
$$

D'où :

$$
\begin{aligned}
\Longrightarrow \ln f(x\,|\theta) &= \underbrace{E_{\mathcal{Z}}[\ln h(x,z\,|\theta)\ |\, x,\theta]}_{=Q_{\mathcal{Z}}(\theta,\theta)} - \underbrace{E_{\mathcal{Z}}[\ln h(z\,|x,\theta)\ |\, x,\theta]}_{=H_{\mathcal{Z}}(\theta,\theta)} \\
\Longrightarrow \ln f(x\,|\theta) &= Q_{\mathcal{Z}}(\theta,\theta) - H_{\mathcal{Z}}(\theta,\theta)
\end{aligned}
$$

Or :

$$
H_{\mathcal{Z}}(\varphi,\theta) - H_{\mathcal{Z}}(\theta,\theta) = \int_{\mathcal{Z}} \left[\ln \frac{k(z\,|x,\varphi)}{k(z\,|x,\theta)}\right] k(z\,|x,\theta)\, dz
$$

Théorème E.4.12 : Inégalité de Jensen
Soit $X \in \mathcal{X}$ une variable aléatoire de densité f, soit $g : \mathcal{X} \longrightarrow \mathbb{R}$ une fonction convexe alors :

$$
E(g(X)) = \int_{\mathcal{X}} g(x)\, f(x)\, dx \leqslant g\left(\int_{\mathcal{X}} f(x)\, dx\right) = g(E(X))
$$

D'après l'inégalité de Jensen, on déduit que, puisque la fonction $t \to -\ln t$ est convexe :

$$
H_{\mathcal{Z}}(\varphi,\theta) - H_{\mathcal{Z}}(\theta,\theta) \leqslant \ln \int_{\mathcal{Z}} \left[\frac{k(z\,|x,\varphi)}{k(z\,|x,\theta)}\right] k(z\,|x,\theta)\, dz = \ln \int_{\mathcal{Z}} k(z\,|x,\varphi)\, dz = \ln 1 = 0
$$

Par conséquent :

$$
\forall \varphi \in \Theta,\ H_{\mathcal{Z}}(\theta,\theta) \geqslant H_{\mathcal{Z}}(\varphi,\theta) \tag{E.44}
$$

De plus :

$$\begin{aligned}
&\text{soit } \varphi \in \Theta \text{ tel que } Q_{\mathcal{Z}}(\varphi, \theta) \geqslant Q_{\mathcal{Z}}(\theta, \theta) \\
\Longrightarrow\ & E_{\mathcal{Z}}[\ln h(x, z|\varphi) \mid x, \theta] - E_{\mathcal{Z}}[\ln h(z|x, \varphi) \mid x, \theta] \geqslant E_{\mathcal{Z}}[\ln h(x, z|\theta) \mid x, \theta] - E_{\mathcal{Z}}[\ln h(z|x, \theta) \mid x, \theta] \\
\Longrightarrow\ & \ln f(x|\varphi) \geqslant \ln f(x|\theta) \qquad \text{(E.45)}
\end{aligned}$$

L'algorithme EM a pour objectif de trouver $\theta^* \in \Theta$ qui maximise $\ln f(x|\theta)$, il est inspiré de ce qui précède :

Algorithme E.4.13 : EM
Les notations adoptées sont celles des paragraphes qui précèdent. L'objectif de cet algorithme est de trouver les paramètres θ qui maximisent la densité $f(x \mid \theta)$:

Etape A : initialisation
On choisit $\theta_0 \in \Theta$ de manière aléatoire.

Etape B : expectation "E"
Pour θ_t, on calcule $Q_{\mathcal{Z}}(\varphi, \theta_t) = \displaystyle\int_{\mathcal{Z}} [\ln h(x, z|\varphi)]\ k(z|x, \theta_t)\, dz$.

Etape C : maximisation "M"
On obtient $\theta_{t+1} = \arg\max_{\varphi \in \Theta} Q_{\mathcal{Z}}(\varphi, \theta_t)$.

Etape D : terminaison
On retourne à l'étape B jusqu'à ce que la suite $(\ln f(x|\theta_t))_t$ converge.

Théorème E.4.14 : convergence de l'algorithme EM
La suite $(\ln f(x|\theta_t))_t$ construite par l'algorthime EM (E.4.13) converge vers un minimum local de la fonction $\theta \longrightarrow f(x \mid \theta)$.

Démonstration (théorème E.4.14) :
Le théorème est démontrée par l'équation (E.45), la suite $(\ln f(x|\theta_t))_t$ est croissante et bornée, donc elle converge. **(E.4.14)** □

E.4.6.2 Exemple : la taille d'un poisson selon le sexe

Soit X une variable aléatoire représentant la taille d'un poisson d'une espèce donnée à l'âge adulte. La taille dépend fortement du sexe du poisson. Soit $Z \in \{0, 1\}$ le sexe du poisson, X est la variable observée, Z la variable manquante. On suppose que, connaissant le sexe du poisson, sa taille suit une loi normale de paramètre $(\mu_i, \sigma_i)_{i \in \{0,1\}}$. Le sexe du poisson suit une loi binomiale de paramètre p.

Dans ce cas, $\theta = (p, \mu_0, \sigma_0, \mu_1, \sigma_1)$, la densité de $X|Z$ est :

$$l(x|z, \theta) = \frac{1}{\sigma_z \sqrt{2\pi}} e^{-\dfrac{(x - \mu_z)^2}{2\sigma_z^2}}$$

et la densité de $Z|\theta$ est :

$$\mathbb{P}(Z = z|\theta) = p\, \mathbf{1}_{\{z=0\}} + (1 - p)\ \mathbf{1}_{\{z=1\}}$$

D'après les hypothèses, on en déduit que :

$$
\begin{aligned}
h(x,z\,|\theta) &= \mathbb{1}_{\{z=0\}}\left[\frac{p}{\sigma_0\sqrt{2\pi}}e^{-\frac{(x-\mu_0)^2}{2\sigma_0^2}}\right]+\mathbb{1}_{\{z=1\}}\left[\frac{(1-p)}{\sigma_1\sqrt{2\pi}}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}\right]\\
\ln h(x,z\,|\theta) &= \mathbb{1}_{\{z=0\}}\left[\ln\frac{p}{\sigma_0\sqrt{2\pi}}-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right]+\mathbb{1}_{\{z=1\}}\left[\ln\frac{(1-p)}{\sigma_1\sqrt{2\pi}}-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right]\\
f(x\,|\theta) &= \frac{p}{\sigma_0\sqrt{2\pi}}e^{-\frac{(x-\mu_0)^2}{2\sigma_0^2}}+\frac{(1-p)}{\sigma_1\sqrt{2\pi}}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}\\
\mathbb{P}(Z=z\,|x,\theta) &= k(z\,|x,\theta)=\frac{h(x,z\,|\theta)}{f(x\,|\theta)}
\end{aligned}
$$

L'objectif est de trouver les véritables valeurs de $(p,\mu_0,\sigma_0,\mu_1,\sigma_1)$ à partir d'une liste d'observation $(x_1,...,x_n)$ donc de trouver :

$$\theta^*=\arg\max_\theta\prod_{i=1}^n f(x_i\,|\theta)=\arg\max_\theta\sum_{i=1}^n\ln f(x_i\,|\theta)$$

Ce problème n'est pas soluble par maximum de vraisemblance. En effet, maximiser la vraisemblance du modèle aboutit à la résolution d'un système d'équations à cinq inconnues insoluble. L'algorithme EM est une alternative, dans ce cas :

$$
\begin{aligned}
Q_{\mathcal{Z}}(\theta,\theta_t) &= \sum_{i=1}^n\sum_{z=0}^1\left[\ln h(x_i,z\,|\theta)\right]\,k(z\,|x,\theta_t)\\
Q_{\mathcal{Z}}(\theta,\theta_t) &= \sum_{i=1}^n\left[\ln\frac{p}{\sigma_0\sqrt{2\pi}}-\frac{(x_i-\mu_0)^2}{2\sigma_0^2}\right]k(0|x_i,\theta_t)+\left[\ln\frac{(1-p)}{\sigma_1\sqrt{2\pi}}-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}\right]k(1|x_i,\theta_t)
\end{aligned}
$$

Trouver $\theta_{t+1}=\arg\max_\theta Q_{\mathcal{Z}}(\theta,\theta_t)$ est un problème plus simple que le précédent et soluble car cette fois, le système d'équation à cinq inconnues obtenu est soluble. Cet exemple est un cas particulier des mélanges de lois normales.

Les formules de Baum-Welch sont un cas particulier de cet algorithme dont la découverte est postérieure. L'algorithme EM est aussi décliné dans plusieurs variantes SEM, SAEM, CEM, ... (voir [Celeux1985], [Celeux1995]). En particulier, comme pour les réseaux de neurones [1], il existe une version stochastique de l'algorithme EM notée SEM.

E.4.7 Démonstration des formules de Baum-Welch avec l'algorithme EM

Démonstration (théorème E.4.3) :

Les formules de Baum-Welch (voir table E.2, page 265) se déduisent de l'algorithme EM (algorithme E.4.13). Soit $M(\phi)$ un modèle de Markov caché dont les paramètres sont le vecteur $\phi=(A_\phi,B_\phi,\pi_\phi,\theta_\phi)$.

1. Annexes : voir paragraphe C.4, page 211

Soit $O = (O_1, \ldots, O_K)$ K séquences d'observations avec $\forall k \in \{1, \cdots, K\}$, $O^k = \left(O_1^k, \ldots, O_{T_k}^k\right)$, $s = (q_1, \ldots, q_T)$ est une séquence d'états cachés. Les densités h, k, f de l'algorithme EM correspondent à :

$$\begin{aligned} h\left(O, s|M\left(\phi\right)\right) &= \prod_{k=1}^{K} \mathbb{P}\left(O_1^k, \ldots, O_{T_k}^k, q_1, \ldots, q_{T_k}|M\left(\phi\right)\right) \\ &= \prod_{k=1}^{K} \pi_{\phi,q_1} b_{\phi,q_1}\left(O_1\right) \left[\prod_{t=2}^{T_k} a_{\phi,q_{t-1}q_t} b_{\phi,q_t}\left(O_t\right)\right] \theta_{\phi,q_{T_k}} \\ f\left(O|M\left(\phi\right)\right) &= \sum_{s} h\left(O, s|M\left(\phi\right)\right) \\ k\left(s|O, M\left(\phi\right)\right) &= \frac{h\left(O, s|M\left(\phi\right)\right)}{f\left(O|M\left(\phi\right)\right)} \end{aligned}$$

On note $\mathcal{S}$ l'ensemble des séquences d'états cachés, alors :

$$Q_{\mathcal{S}}\left(\psi, \phi\right) = \sum_{s \in \mathcal{S}} \ln h\left(O, s|M\left(\psi\right)\right) k\left(s|O, M\left(\phi\right)\right) \tag{E.46}$$

$$= \sum_{s \in \mathcal{S}} k\left(s|O, M\left(\phi\right)\right) \sum_{k=1}^{K} \left[\begin{array}{ll} & \ln \pi_{\psi,q_1} + \ln b_{\psi,q_1}\left(O_1\right) \\ + & \sum\limits_{t=2}^{T_k} \ln a_{\psi,q_{t-1}q_t} + \ln b_{\psi,q_t}\left(O_t\right) \\ + & \ln \theta_{\psi,q_{T_k}} \end{array}\right] \tag{E.47}$$

Θ est l'ensemble des paramètres $\phi = (A_\phi, B_\phi, \pi_\phi, \theta_\phi)$ vérifiant les contraintes inhérentes aux chaînes de Markov cachées. L'objectif est de trouver :

$$\psi^* = \arg\max_{\psi \in \Theta} Q_{\mathcal{S}}\left(\psi, \phi\right)$$

Pour cela, on écrit différemment l'équation (E.47) :

$$Q_{\mathcal{S}}\left(\psi, \phi\right) = \sum_{n=1}^{N} C_n \ln \pi_{\psi,n} + \sum_{n=1}^{N}\sum_{m=1}^{N} A_{nm} \ln a_{\psi,nm} + \sum_{n=1}^{N}\sum_{o} B_{n,o} \ln b_{\psi,n}\left(o\right) + \sum_{n=1}^{N} D_n \ln \theta_{\psi,n} \tag{E.48}$$

Les matrices $C_n, A_{nm}, B_{n,o}, D_n$ ont les valeurs données par les équations (E.40) à (E.43) (page 276).

(E.4.3) □

E.4.8 Amélioration de l'apprentissage

Les formules de Baum-Welch (voir table E.2) impliquent que si un coefficient devient nul après un certain nombre d'itérations, il le restera aux itérations suivantes. Cela ne signifie pas que la valeur optimale pour ce coefficient soit différente de zéro, cependant si ce n'est pas le cas, l'apprentissage est biaisé. C'est pourquoi on préfère que tous les coefficients soient non nuls.

Afin d'éviter cet écueil, les coefficients inférieurs à un certain seuil sont modifiés aléatoirement de sorte qu'ils soient supérieurs à ce seuil, ce seuil s_t décroît avec le nombre d'itérations t :

$$s_t = \frac{s_0}{1+\gamma t} \text{ avec } \gamma \in \mathbb{R}^+$$

La vraisemblance des modèles obtenue lors des itérations successives décroît globalement mais l'aléatoire introduit entre chaque itération fait osciller la valeur de cette vraisemblance lorsque les coefficients sont proches d'un optimum local. Cette oscillation décroît au fur et à mesure que s_t décroît.

E.5 Observations continues et réseau de neurones

Jusqu'à présent, les observations étaient discrètes : les séquences d'observations étaient des séquences d'entiers pris dans un ensemble fini. Lorsque les données observées sont continues, il est possible de se ramener à ce cas-là en construisant une partition de l'espace (avec l'algorithme des centres mobiles par exemple, voir paragraphe H.1). Néanmoins, une classification traite de manière insatisfaisante les ambiguïtés : les observations situées sur une frontière entre deux classes (voir figure E.8). Il est alors préférable de conserver des probabilités d'appartenir à telle ou telle classe (voir paragraphe H.1.3).

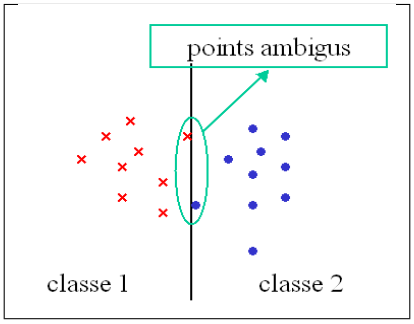


Fig. E.8: Ambiguïtés d'un partitionnenement d'un espace continu.

E.5.1 Chaîne de Markov cachée incluant un réseau de neurones

Les chaînes de Markov cachées incluant un réseau de neurones sont qualifiées de *hybrides*, les émissions sont en quelque sorte sous-traitées par la chaîne de Markov à un réseau de neurones.

E.5.1.1 Initialisation

On suppose que l'espace des observations continues a été partitionné à l'aide des méthodes présentées dans les paragraphes H.1.1, H.1.2, H.1.3 (pages 346 et suivantes), on dispose donc pour chaque observation x des probabilités $\mathbb{P}(c\,|x\,)$, elles sont retournées par un réseau de neurones classifieur (voir paragraphe C.5, page 217) dont l'apprentissage est évoqué au paragraphe H.1.3.

E.5.1.2 Des observations discrètes aux observations continues

Jusqu'à présent, les modèles d'émissions ont toujours été discrets, les modèles de Markov retournaient la probabilités de séquences discrètes. Les modèles d'émissions discretes sont entièrement décrits par une matrice :

$$(b_{i,o})_{\substack{1\leqslant i\leqslant N\\1\leqslant o\leqslant O}} = (\mathbb{P}(o\,|i\,))_{\substack{1\leqslant i\leqslant N\\1\leqslant o\leqslant O}}$$

avec
- N est le nombre d'états de la chaîne de Markov
- O le nombre d'observations possibles
- $\mathbb{P}(o\,|i\,)$ est la probabilité d'émettre l'observation o connaissant l'état i

Comme les observations sont continues, il faut construire un modèle d'émission estimant la densité d'une observation continue x sachant l'état i : $f(x\,|i\,)$. On note $x \longrightarrow Rn(x) = (\mathbb{P}(c\,|x\,))_{1\leqslant c\leqslant C}$ la fonction définie par le réseau de neurones appris grâce à la classification (voir paragraphe H.1.3).

Définition E.5.1 : Chaîne de Markov cachée hybride
Une chaîne de Markov cachée hybride dont les observations sont continues vérifie les conditions 2 à 4 vérifiées par une chaîne de Markov cachée dont les observations sont discrètes (voir définition E.2.1, page 249), et les conditions 1 à 3 qui suivent :

1. La densité d'une observation ne dépend que de sa classe : $f(x\,|c,i\,) = f(x\,|c\,)$
2. La probabilité que l'observation à l'instant t soit dans la classe c ne dépend que de l'état à l'instant t :

$$\mathbb{P}(O_t \in classe(c)\ |\ q_1,...,q_t,O_1,...,O_{t-1}\,) = \mathbb{P}(O_t \in classe(c)\,|q_t\,) = \mathbb{P}(c\,|q_t\,)$$

3. La probabilité que l'observation à l'instant t soit dans la classe c ne dépend pas du temps :

$$\forall t_1,t_2,\forall i,c,\ \mathbb{P}(O_{t_1} \in classe(c)\,|q_{t_1}=i\,) = \mathbb{P}(O_{t_2} \in classe(c)\,|q_{t_2}=i\,)$$

Propriété E.5.2 : probabilité d'émission
On en déduit que la densité $f(x|i)$ d'une observation x sachant l'état i est :

$$f(x\,|i\,) = \sum_{c=1}^{C} f(x,c\,|i\,) = \sum_{c=1}^{C} f(x\,|c,i\,)\,\mathbb{P}(c\,|i\,) = \sum_{c=1}^{C} \frac{\mathbb{P}(c\,|x\,)\,\mathbb{P}(c\,|i\,)\,f(x)}{\mathbb{P}(c)}$$

où :

- $\mathbb{P}(c\,|x\,)$ est estimé par le réseau de neurones : $\mathbb{P}(c\,|x\,) = Rn(x)$
- $\mathbb{P}(c\,|i\,)$ est un coefficient qui sera estimé de la même manière que les probabilités de transition
- $f(x)$ est la densité des observations, elle est inconnue mais peut être estimée par (H.5)
- $\mathbb{P}(c)$ est la probabilité de la classe c, elle est incoonue mais peut être estimée par (H.4)

On note $\mathbb{P}(c|i)_{i,c} = (c_{i,c})_{i,c}$ la matrice des probabilités émissions dans le cas d'observations continues.

E.5.2 Réestimation de $(c_{i,c})_{i,c}$

On définit de nouveaux coefficients pour le modèle de Markov caché :

$$(c_{ci})_{\substack{1\leqslant i\leqslant N\\1\leqslant c\leqslant C}} = (\mathbb{P}(c|i))_{\substack{1\leqslant i\leqslant N\\1\leqslant c\leqslant C}}$$

$$\boxed{\overline{c_{i,c}} = \mathbb{P}\left(c \,|i, O\right) = \frac{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \frac{\beta_i^k\left(t\right) \mathbb{P}\left(O_t^k \,|c\right) c_{i,c} \alpha_i^k\left(t\right)}{\displaystyle\sum_{d=1}^{N} \mathbb{P}\left(O_t^k \,|c\right) c_{i,d}}}{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_i^k\left(t\right) \beta_i^k\left(t\right)}}$$

Tab. E.3: Formules de réestimation de Baum-Welch, modèle hybride

De la même manière que pour les formules de Baum-Welch, on cherche à estimer $\mathbb{P}\left(c, i, O\right)$ où O est une séquence d'observations. C_t désigne la classe de l'observation O_t.

$$\mathbb{P}\left(c \,|i\right) = \overline{c_{i,c}} = \frac{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \mathbb{P}\left(C_t = c,\, q_t = i, O^k\right)}{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \mathbb{P}\left(q_t = i,\, O^k\right)}$$

La démonstration de la formule de réestimation de $c_{i,c}$ pour un modèle apprenant la séquence d'observations $(O_1, \ldots, O_T)$:

$$\begin{aligned}
\mathbb{P}\left(C_t, q_t, O\right) &= \mathbb{P}\left(O_{t+1}, .., O_T \,|C_t, q_t, O_1, ..., O_t\right) \mathbb{P}\left(C_t, q_t, O_1, ..., O_t\right) \\
\mathbb{P}\left(C_t, q_t, O\right) &= \mathbb{P}\left(O_{t+1}, .., O_T \,|q_t\right) \mathbb{P}\left(O_t \,|C_t, q_t, O_1, ..., O_{t-1}\right) \mathbb{P}\left(C_t, q_t, O_1, ..., O_{t-1}\right) \\
\mathbb{P}\left(C_t, q_t, O\right) &= \beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right) \mathbb{P}\left(C_t \,|q_t, O_1, ..., O_{t-1}\right) \mathbb{P}\left(q_t, O_1, ..., O_{t-1}\right) \\
\mathbb{P}\left(C_t, q_t, O\right) &= \beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right) \mathbb{P}\left(C_t \,|q_t\right) \sum_{q_{t-1}} \mathbb{P}\left(q_t, q_{t-1}, O_1, ..., O_{t-1}\right) \\
\mathbb{P}\left(C_t, q_t, O\right) &= \beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right) \, c_{q_t, C_t} \sum_{q_{t-1}} a_{q_{t-1}, q_t} \, \alpha_{q_{t-1}}\left(t\right) = \frac{\beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right) \, c_{q_t, C_t} \, \alpha_{q_t}\left(t\right)}{b_{q_t}\left(O_t\right)} \\
\mathbb{P}\left(C_t, q_t, O\right) &= \frac{\beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right) \, c_{q_t, C_t} \, \alpha_{q_t}\left(t\right)}{\displaystyle\sum_{d=1}^{N} \mathbb{P}\left(O_t \,|C_t = d\right) \, c_{q_t, d}} \qquad \text{(E.49)} \\
\text{Rappel :} \mathbb{P}\left(q_t, O\right) &= \alpha_{q_t}\left(t\right) \beta_{q_t}\left(t\right)
\end{aligned}$$

Si le modèle apprend plusieurs séquences $\left(O^1, \ldots, O^K\right)$ de longueurs respectives $(T_1, \ldots, T_K)$, alors la formule de la table E.3 vient s'ajouter à celles de la table E.2.

E.5.3 Réestimation des $\mathbb{P}\left(c|o\right)$

Ces probabilités sont fournies par le réseau de neurones dont l'apprentissage peut être mis en parallèle avec celui du modèle de Markov caché ou être différé. Dans cette seconde solution, il faut estimer les probabilités $\left(\mathbb{P}\left(c \,|x\right)\right)_{1 \leqslant c \leqslant C}$ qui diminueront la vraisemblance des observations. De la même manière que précédemment, on estime la probabilité $\mathbb{P}\left(C_t \,\middle|O_t^k\right)$ pour la séquence O^k.

C_t désigne toujours la classe de l'observation O_t, la démonstration des formules sera faite pour une séquence, pour abréger les notations, $O = O^k$:

$$\boxed{\begin{array}{l} i^\circ \text{ ligne de } X : X_i = O_t^k \\ \\ k^\circ \text{ ligne de } Y : Y_k = \left(\dfrac{1}{\mathbb{P}\left(O^k\right)} \displaystyle\sum_{q_t} \dfrac{\beta_{q_t^k}\left(t\right) \mathbb{P}\left(O_t^k \,|c\right)\; c_{q_t,c}\, \alpha_{q_t^k}\left(t\right)}{\displaystyle\sum_{d=1}^{C} \mathbb{P}\left(O_t^k \,|d\right)\; c_{q_t,d}} \right)_{1 \leqslant c \leqslant C} \end{array}}$$

Tab. E.4: Formules de réestimation de Baum-Welch, modèle hybride, partie réseau de neurones. On passe d'une ligne à la suivante en incrémentant t ou lorsque t correspond à la dernière observations de la séquence O^k, en incrémentant k. Les matrices X et Y constituent la base d'apprentissage du réseau de neurones, X contient les entrées, Y les sorties désirées.

$$\boxed{\overline{Y_{tc}^k} = \frac{1}{\mathbb{P}\left(O^k\right)} \left(\sum_{q_t} \frac{\beta_{q_t}^k\left(t\right)\; Y_{tc}^k\, c_{q_t,c}\, \alpha_{q_t}^k\left(t\right)}{\displaystyle\sum_{d=1}^{C} Y_{td}^k\, c_{q_t,d}} \right)}$$

Tab. E.5: Formules de réestimation de Baum-Welch, modèle hybride, partie réseau de neurones. Cette formule de réestimation vient en complément de la table E.4 où le terme Y_i est remplacé par le vecteur $\left(\overline{Y_{t1}^k}, \ldots, \overline{Y_{tC}^k}\right)$ obtenu après convergence de la probabilité $\mathbb{P}\left(O_t \mid M\right)$ et en utilisant la formule de réestimation ci-dessus. La première valeur pour Y_{tC}^k correspond à la sortie du réseau de neurones avant réapprentissage.

$$\begin{aligned} \mathbb{P}\left(C_t|O_t\right) = \mathbb{P}\left(C_t|O\right) &= \frac{\mathbb{P}\left(C_t, O\right)}{\mathbb{P}\left(O\right)} = \frac{1}{\mathbb{P}\left(O\right)} \sum_{q_t} \mathbb{P}\left(C_t, q_t, O\right) \\ &= \frac{1}{\mathbb{P}\left(O\right)} \sum_{q_t} \frac{\beta_{q_t}\left(t\right) \mathbb{P}\left(O_t \,|C_t\right)\; c_{q_t,C_t}\, \alpha_{q_t}\left(t\right)}{\displaystyle\sum_{d=1}^{N} \mathbb{P}\left(O_t \,|C_t = d\right)\; c_{q_t,d}} \quad \text{d'après (E.49)} \end{aligned}$$

Par conséquent, dans un premier temps, la base d'apprentissage du réseau de neurones est la base (X, Y) définies par la table E.4.

Toutefois la formule décrite dans cette table E.4 ne donne pas le vecteur de probabilité Y qui maximise la vraisemblance des observations mais celui-ci peut être obtenu en adaptant l'algorithme EM à ce cas-là. On note $Y_{ct} = \mathbb{P}\left(O_j \mid C_c\right)$. O_t désigne la $t^{\text{ème}}$ observations de la séquence $O = (O_1, \ldots, O_T)$ et C_i la $c^{\text{ème}}$ classe. Par conséquent Y_{tc} est la sortie $c^{\text{ème}}$ désirée du réseau de neurones lorsqu'il a pour entrée le vecteur O_t. Comme les coefficients $c_{i,c}$ (voir table E.3), les nombres Y_{tc} sont mis à jour selon la formules de la table E.5.

Il faut d'ajouter que l'utilisation de telles formules de convergence mènent souvent à des sorties désirées pour le réseau de neurones qui sont soient nulles, soient égales à un. La remarque C.5.6 [2] suggère de ne

2. Annexes : voir paragraphe C.5.6, page 223

pas utiliser ces sorties telles quelles afin de faciliter l'apprentissage.

Algorithme E.5.3 : apprentissage alterné du modèle hybride complet
L'apprentissage proposé alterne l'apprentissage de la chaîne de Markov cachée de celui du réseau de neurones, il est constitué de trois étapes :

Etape A : initialisation
Initialisation du réseau de neurones à l'aide des méthodes proposées dans les paragraphes : H.1.1 (page 346), H.1.2 (page 348), H.1.3 (page 349)

Etape B : apprentissage MMC
Apprentissage Baum-Welch des probabilités de transitions et d'émissions, voir les paragraphes : E.2 (page 265), E.5.2 (page 282),

Etape C : apprentissage RN
Réapprentissage du réseau de neurones, voir ce paragraphe E.5.3 (page 283) ainsi que les tables E.4 et E.5.
Retour à l'étape B jusqu'à convergence.

Remarque E.5.4: convergence non monotone
Il est conseillé de conserver les versions des modèles à chaque itération car la convergence est rarement monotone.

E.5.4 Emissions continues modélisées par une loi normale multidimensionnelle

Les probabilités d'émission peuvent être modélisées par des lois normales, soit $(O_1, \ldots, O_T)$ une séquence d'observations et $(q_1, \ldots, q_T)$ une séquence d'états du modèle M. On suppose que la variable :

$$O_t | q_t = i \sim \mathcal{N}(\mu_i, V_i)$$

Par conséquent :

$$b_i(O_t) = f(O_t | q_t = i, M) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{\det(V_i)}} e^{-\frac{1}{2}(O_t - \mu_i)' V_i^{-1} (O_t - \mu_i)}$$

Les formules de réestimation (voir [Bottou1991]) à utiliser lors de l'algorithme de Baum-Welch sont les suivantes pour les séquences d'observations $\left(O_1^k, \ldots, O_{T_k}^k\right)_{1 \leqslant k \leqslant K}$, on note $P_k = f\left(O^k | M\right)$ où f est la densité des séquences d'observations :

$$\overline{\mu_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) O_t^k}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i)} \tag{E.50}$$

$$\overline{V_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) O_t^k O_t^{k\prime}}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i)} - \mu_i \mu_i' \tag{E.51}$$

L'inconvénient de ces modèles est le calcul de la densité qui implique un produit matrice en $O\left(d^3\right)$ où d est la dimension de l'espace des observations. Etant donné la taille considérable de cette matrice, elles

sont soit réduites à leur diagonale, soit factorisées entre les états. Pour cette dernière solution, le modèle hybride résultant est agencé de manière semblable à celui regroupant un modèle de Markov caché et un réseau de neurones. Le réseau de neurones agit comme un classifieur commun à tous les états, dans l'autre cas, c'est un mélange de lois normales qui modélisent la distribution des observations.

Remarque E.5.5: calcul pratique de probabilités : utlisation de coûts
L'utilisation de lois normales peut poser des problèmes informatiques de mise en œuvre. En effet, les probabilités sont alors souvent très faibles pour des espaces de grandes dimensions, quelques dizaines comme pour la reconnaissance de l'écriture manuscrite. Il arrive fréquemment que l'estimation de tels modèles nécessite le calcul de probabilités parfois inférieures à 10^{-300} qui est la limite d'un réel codé informatique sur huit octets. Il est alors préférable d'utiliser des coûts ou logarithme de probabilités lors des calculs.

E.6 Chaînes de Markov d'ordres supérieurs

Jusqu'à présent, seules les chaînes de Markov cachées d'ordre un ont été utilisées, ceci signifie que l'état à l'instant t ne dépend que de l'état à l'instant $t-1$, un ordre supérieur signifie que l'état à l'instat t dépend de plusieurs des états précédents.

E.6.1 Définition d'une chaîne de Markov d'ordre n

La définition suivante concerne une chaîne de Markov et non une chaîne de Markov cachée : les émissions ne sont pas prises en compte.

Définition E.6.1 : Chaîne de Markov d'ordre n
Soit $N \in \mathbb{N}^*$, on appelle une chaîne de Markov à N états d'ordre n une chaîne de Markov qui vérifie les conditions suivantes :

1. l'état à l'instant t ne dépend que des états aux instants $(t-1, ..., t-n)$. Par conséquent :

$$\forall t > n,\ \mathbb{P}\left(q_t \,|q_{t-1}, ..., q_1, M\right) = \mathbb{P}\left(q_t \,|q_{t-1}, ..., q_{t-n}, M\right)$$

2. les probabilités de transition ne dépendent pas du temps, par conséquent :

$$\forall t_1, t_2 > 1, \forall \left(i_0, i_1, ..., i_d\right),$$
$$\mathbb{P}\left(q_{t_1} = i_0 \,|q_{t_1-1} = i_1, ..., q_{t_1-d} = i_d, M\right) = \mathbb{P}\left(q_{t_2} = i_0 \,|q_{t_2-1} = i_1, ..., q_{t_2-d} = i_d, M\right)$$

On note $\mathcal{M}_n$ l'ensemble des chaîne de Markov d'ordre n.

E.6.2 Descente d'ordre

Tout d'abord, on définit la fonction suivante f :

$$\begin{aligned} &g : \{1, ..., N\}^n \longrightarrow \{1, ..., N^n\} \\ &g\left(i_1, ..., i_n\right) = \sum_{k=1}^{n} \left(i_k - 1\right) N^{k-1} + 1 \end{aligned} \tag{E.52}$$

Propriété E.6.2 : homomorphisme
Si g est la fonction définie en (E.52), alors g est bijective.

Par conséquent, g^{-1} existe et, on notera $\left[g^{-1}(l)\right]_k$ la k° coordonnées de $g^{-1}(l)$. Cette fonction est tout simplement l'écriture des entiers en base N.

Dans toute la suite, les état de sorties et d'entrées ne seront plus distincts des autres états, ceci permettra de ne pas traiter les probabilités de transition, les probabilités d'entrée et les probabilités de sortie de manière séparée.

Soit une chaîne de Markov M d'ordre n contenant N états représentés par l'ensemble $(1,\ldots,N)$. Cette chaîne est entièrement définie par une hyper-matrice carrée $A_M \in \mathcal{M}_N^{n+1}(\mathbb{R})$ contenant les $(N)^{n+1}$ probabilités de transition de la chaîne de Markov $M \in \mathcal{M}_n$:

$$A_M(i_1,...,i_n,i_{n+1}) = \mathbb{P}(q_t = i_{n+1} \,|q_{t-1} = i_n,...,q_{t-n} = i_1, M)$$

Si e est l'état d'entrée du modèle, on pose comme convention :

$$A_M(i_1,\ldots,i_{n+1}) = \begin{cases} 0 \text{ si } i_{n+1} = e \\ 0 \text{ si } \exists k \leqslant n \text{ tel que } i_k \neq e \text{ et } i_{k+1} = e \\ 1 \text{ si } \exists k \text{ tel que } 3 \leqslant k \leqslant n \text{ et } \forall k' < k,\ i_{k'} = e \text{ et } \forall k' \geqslant k,\ i_{k'} \neq e \\ \mathbb{P}(q_t = i_{n+1} \,|\, q_{t-1} = i_n,\ldots,q_{t-n} = i_1) \text{ sinon} \end{cases} \tag{E.53}$$

On construit la chaîne de Markov M' d'ordre un contenant N^n états représentés par l'ensemble $(1,\ldots,N^n)$. Cette chaîne est entièrement définie par sa matrice carrée $A'_{M'} \in \mathcal{M}_{N^n}^2(\mathbb{R})$ contenant les $(N^n)^2$ probabilités de transition de la chaîne $M' \in \mathcal{M}_1$:

$$A'_{M'}(k,l) = \mathbb{P}\left(q_t = l \,\middle|\, q_{t-1} = k, M'\right)$$

La matrice des transitions $A'_{M'}$ est définie à partir de l'hyper-matrice A_M :

$$A'_{M'}(k,l) = \begin{cases} A_M\left(\left[g^{-1}(k)\right]_1, g^{-1}(l)\right) \text{ si } \forall i \in \{1,...,n-1\},\ \left[g^{-1}(l)\right]_i = \left[g^{-1}(k)\right]_{i+1} \\ 0 \text{ sinon} \end{cases} \tag{E.54}$$
avec g la fonction définie en (E.52)

Enfin on définit la fonction h :

$$\begin{array}{l} h : \mathcal{M}_n \longrightarrow \mathcal{M}_1 \\ h(M) = M' \text{ avec } M' \text{ construite comme en (E.54)} \end{array} \tag{E.55}$$

On doit d'abord définir l'équivalence entre deux chaînes de Markov :

Définition E.6.3 : équivalence entre deux chaînes de Markov
Soient M_1 et M_2 deux chaînes de Markov, on note S l'ensemble des séquences d'états, alors :

$$(M_1 \Longleftrightarrow M_2) \Longleftrightarrow (\forall s \in S,\ \mathbb{P}(s|M_1) = \mathbb{P}(s|M_2))$$

On peut maintenant énoncer le théorème suivant :

Théorème E.6.4 : homomorphisme
Avec ces notations, la fonction h (E.55) définit un homomorphisme de $(\mathcal{M}_n, \Longleftrightarrow)$ dans $(\mathcal{M}_1, \Longleftrightarrow)$ où $\Longleftrightarrow$ est la relation d'équivalence entre deux chaînes de Markov.

Rappel :

Définition E.6.5 : homomorphisme
Soit $(E, \perp)$ et $(F, \perp)$ deux espaces munis chacun d'une relation d'équivalence.
Soit $h : (E, \perp) \longrightarrow (F, \perp)$ une fonction, h est un homomorphisme si et seulement si :

$$\forall\, (x, y) \in E^2,\ x \perp y \Longrightarrow h\,(x) \perp h\,(y)$$

Démonstration (theoreme E.6.4) :

Rappel : L'état 0 correspond à l'état d'entrée.

Soit $s = (s_1, \ldots, s_T)$ une séquence d'états du modèle M.
On définit la séquence d'états $u = k\,(s)$ comme suit :

$$u = k\,(s) = (u_1, ..., u_T) = \left(\begin{pmatrix} u_{11} \\ \vdots \\ u_{1n} \end{pmatrix}, ..., \begin{pmatrix} u_{T1} \\ \vdots \\ u_{Tn} \end{pmatrix} \right) \in [\{1, ..., N\}^n]^T$$

avec :

$$\forall t \in \{1, \cdots, T\},\ \forall l \in \{1, \cdots, n\},\ u_{tl} = \begin{cases} s_{t+l-n} \text{ si } t + l - n > 0 \\ e \text{ si } l + t - n \leqslant 0 \text{ où } e \text{ est l'état d'entrée de } M \end{cases}$$

Cette séquence vérifie :

$$\begin{array}{rrcl} \forall t \in \{2, \cdots, T\},\ \forall l \in \{1, \cdots, n-1\}, & u_{t-1,l+1} & = & u_{tl} \\ \forall t \in \{1, \cdots, T-1\}, & u_{tn} & = & u_{t+1,1} \end{array}$$

La convention choisie en (E.53) implique que :

$$\mathbb{P}\,(s\,|M\,) = \mathbb{P}\left(u\,\middle|M'\right) = \mathbb{P}\left(h\,(s)\,\middle|M'\right)$$

Donc, soit $(M, N) \in (\mathcal{M}_n)^2$,

$$\Big[M \Longleftrightarrow N\Big] \Longrightarrow \Big[\forall s \in S,\ {}_p rs\,|M = \mathbb{P}\,(s\,|N\,) \Longrightarrow \forall s,\, \mathbb{P}\,(k\,(s)\,|h\,(M)\,) = \mathbb{P}\,(k\,(s)\,|g\,(N)\,)\Big]$$

De plus, d'après (E.54), on déduit que :

$$\forall u,\ \Big[\nexists s \in S \text{ tel que } k\,(s) = u\Big] \Longrightarrow \mathbb{P}\,(u\,|g\,(M)\,) = \mathbb{P}\,(u\,|g\,(N)\,) = 0$$

Donc :

$$\Big[M \Longleftrightarrow N\Big] \Longrightarrow \Big[g\left(M\right) \Longleftrightarrow g\left(N\right)\Big]$$

(E.6.4) □

E.6.3 Définition d'une chaîne de Markov cachée d'ordre n

Le paragraphe précédent a montré comment transformer une chaîne de Markov d'ordre n en une chaîne de Markov d'ordre un. Ce résultat peut être étendu aux chaînes de Markov cachées :

Définition E.6.6 : chaîne de Markov cachée d'ordre n
Soit $N \in \mathbb{N}^*$, soit une chaîne de Markov cachée à N états d'ordre n dont les émissions sont discrètes et appartiennent à l'ensemble $(1, \ldots, D)$, cette chaîne vérifie les hypothèses suivantes :

1. L'état à l'instant t ne dépend que des états aux instants $(t-1, ..., t-n)$. Par conséquent :

$$\forall t > n, \quad \mathbb{P}\left(q_t \left| q_{t-1}, ..., q_1, M\right.\right) = \mathbb{P}\left(q_t \left| q_{t-1}, ..., q_{t-n}, M\right.\right)$$

2. Les probabilités de transition ne dépendent pas du temps, par conséquent :

$$\begin{array}{l}\forall t_1, t_2 > 1, \ \forall \left(i_0, i_1, ..., i_n\right), \\ \mathbb{P}\left(q_{t_1} = i_0 \left| q_{t_1-1} = i_1, ..., q_{t_1-n} = i_n, M\right.\right) = \mathbb{P}\left(q_{t_2} = i_0 \left| q_{t_2-1} = i_1, ..., q_{t_2-n} = i_n, M\right.\right)\end{array}$$

3. Les probabilités d'émission ne dépendent que des états aux instants $(t, ..., t-n+1)$:

$$\forall t \geqslant 1, \mathbb{P}\left(O_t \left| q_1, ..., q_t, O_1, ..., O_{t-1}, M\right.\right) = \mathbb{P}\left(O_t \left| \left(q_t, \ldots, q_{t-n+1}\right), M\right.\right)$$

4. Les probabilités d'émission ne dépendent pas du temps :

$$\begin{array}{l}\forall t_1, t_2 > 1, \ \forall \left(i_1, ..., i_n\right), \ \forall o, \\ \mathbb{P}\left(O_{t_1} = o \left| q_{t_1} = i_1, ..., q_{t_1-n+1} = i_n, M\right.\right) = \mathbb{P}\left(O_{t_2} = o \left| q_{t_2} = i_1, ..., q_{t_2-n+1} = i_n, M\right.\right)\end{array}$$

Remarque E.6.7: autres types d'émission
Cette définition peut être déclinée pour des observations d'un type différent (réseau de neurones, normales...).

On note $\mathcal{C}_n$ l'ensemble des chaînes de Markov cachées d'ordre n, en appliquant les résultats du paragraphe précédent, il est possible de construire une fonction h' :

$$\begin{array}{l} h' : \mathcal{C}_n \longrightarrow \mathcal{C}_1 \\ h'\left(C\right) = C' \text{ avec } M_{C'} = h\left(M_C\right) \text{ où} \\ \qquad\qquad M_C \text{ est la chaîne de Markov cachée dans } C \\ \qquad\qquad M_{C'} \text{ est la chaîne de Markov cachée dans } C' \end{array} \tag{E.56}$$

$h\left(C\right)$ vérifie :

$$\mathbb{P}\left(q_t = l \left| q_{t-1} = k, M'\right.\right) = \begin{cases} \mathbb{P}\left(q_t = \left[g^{-1}(l)\right]_n \left| q_{t-1} = \left[g^{-1}(k)\right]_n, \ldots, q_{t-n} = \left[g^{-1}(k)\right]_1, M\right.\right) \\ \qquad \text{si } \forall i \in \{1, ..., n-1\},\ \left[g^{-1}(l)\right]_i = \left[g^{-1}(k)\right]_{i+1} \\ e \text{ sinon } (e \text{ est l'état d'entrée de la chaîne de Markov}) \end{cases}$$

$$\mathbb{P}\left(O_t = o \left| q_t = l, M'\right.\right) = \mathbb{P}\left(O_t = o \left| q_t = \left[g^{-1}(l)\right]_n, \ldots, q_{t-n+1} = \left[g^{-1}(l)\right]_1, M\right.\right)$$

On doit d'abord définir l'équivalence entre deux chaînes de Markov cachées :

Définition E.6.8 : équivalence entre deux chaînes de Markov cachées
Soient C_1 et C_2 deux chaînes de Markov cachée, on note $\mathcal{O}$ l'ensemble des séquences d'observations, alors :

$$\Big[C_1 \Longleftrightarrow C_2\Big] \Longleftrightarrow \Big[\forall O \in \mathcal{O},\ \mathbb{P}\left(O|C_1\right) = \mathbb{P}\left(O|C_2\right)\Big]$$

On peut maintenant énoncer le théorème suivant :

Théorème E.6.9 : homomorphisme
Avec ces notations, la fonction h' (E.56) définit un homomorphisme de $(\mathcal{C}_n, \Longleftrightarrow)$ dans $(\mathcal{C}_1, \Longleftrightarrow)$ où $\Longleftrightarrow$ est la relation d'équivalence entre deux chaînes de Markov. De plus, $\forall C \in \mathcal{C}_n,\ h(C) \Longleftrightarrow C$.

Démonstration (théorème E.6.9) :
Ce théorème est un corollaire du théorème E.6.4 (page 288). **(E.6.9)** □

E.6.4 Définition d'une chaîne de Markov cachée d'ordre (p, q)

Ces modèles sont ceux dont les hypothèses sont les moins contraignantes.

Définition E.6.10 : chaîne de Markov cachée d'ordre (p, q)
Soit $N \in \mathbb{N}^*$, soit une chaîne de Markov cachée à N états d'ordre $p > 0$ dont les émissions sont discrètes et appartiennent à l'ensemble $(1, \ldots, D)$, cette chaîne vérifie les hypothèses suivantes :

1. L'état à l'instant t ne dépend que des états aux instants $(t-1, ..., t-p)$ et des observations aux instants $(O_{t-1}, \ldots, O_{t-q})$:

$$\forall t > p,\ \mathbb{P}\left(q_t \left| \overline{q_{t-1}}, \overline{O_{t-1}}, M\right.\right) = \mathbb{P}\left(q_t \left| q_{t-1}, ..., q_{t-p}, O_{t-1}, ..., O_{t-q}, M\right.\right)$$

2. Les probabilités de transition ne dépendent pas du temps, Par conséquent :

$$\begin{array}{l}\forall t_1, t_2 > 1,\ \forall \left(i_0, ..., i_p\right),\ \forall \left(x_1, ..., x_q\right), \\ \mathbb{P}\left(q_{t_0} = i_0 \left| q_{t_1} = i_1, ..., q_{t_1-p+1} = i_n, O_{t_1} = x_1, ..., O_{t_1-q} = x_n, M\right.\right) \\ \quad = \mathbb{P}\left(q_{t_2} = i_0 \left| q_{t_2} = i_1, ..., q_{t_2-n+1} = i_n, O_{t_2} = x_1, ..., O_{t_2-q} = x_n, M\right.\right)\end{array}$$

3. Les probabilités d'émission ne dépendent que des états aux instants $(t, ..., t-p+1)$ et des observations aux instants $(O_{t-1}, \ldots, O_{t-Q})$:

$$\forall t \geqslant 1,\ \mathbb{P}\left(O_t \left| q_1, ..., q_t, O_1, ..., O_{t-1}, M\right.\right) = \mathbb{P}\left(O_t \left| \left(q_t, \ldots, q_{t-p+1}\right), \left(O_{t-1}, \ldots, O_{t-q}\right), M\right.\right)$$

4. Les probabilités d'émission ne dépendent pas du temps :

$$\begin{array}{l}\forall t_1, t_2 > 1,\ \forall \left(i_1, ..., i_p\right),\ \forall \left(x_1, ..., x_q\right),\ \forall o, \\ \mathbb{P}\left(O_{t_1} = o \left| q_{t_1} = i_1, ..., q_{t_1-p+1} = i_n, O_{t_1} = x_1, ..., O_{t_1-q} = x_n, M\right.\right) \\ \quad = \mathbb{P}\left(O_{t_2} = o \left| q_{t_2} = i_1, ..., q_{t_2-n+1} = i_n, O_{t_2} = x_1, ..., O_{t_2-q} = x_n, M\right.\right)\end{array}$$

Grâce à une démonstration similaire, les paragraphes précédents nous permettent d'énoncer le théorème suivant :

Théorème E.6.11 : homomorphisme
Soit $\mathcal{C}_{p,q}\left(\mathcal{X}\right)$ l'ensemble des chaînes de Markov cachées d'ordre (p, q) dont les observations appartiennent à l'ensemble $\mathcal{X}$. Alors il existe un homomorphisme de $\left(\mathcal{C}_{p,q}\left(\mathcal{X}\right), \Longleftrightarrow\right)$ dans $\left(\mathcal{C}_{1,1}\left(\mathcal{X}^q\right), \Longleftrightarrow\right)$

Démonstration (théorème E.6.11) :
La démonstration est similaire à celle du théorème E.6.4 (page 288). **(E.6.11)** □

Corollaire E.6.12 : homomorphisme
Si $\mathcal{X}$ est un ensemble fini, il existe un homomorphisme de $\left(\mathcal{C}_{p,q}\left(\mathcal{X}\right), \Longleftrightarrow\right)$ dans $\left(\mathcal{C}_{1,0}\left(\mathcal{X}^q\right), \Longleftrightarrow\right)$

E.6.5 Conclusion

Cette annexe a présenté les modèles de Markov cachés, leur utilisation pour la modélisation de séquences discrètes ou continues et leur estimation. Pour des raisons calculatoires, les chaînes de Markov cachées

d'ordre (p, q) avec $q > 0$ aux émissions continues sont peu utilisées. Il est alors possible de n'envisager que des modèles de chaînes de Markov cachées d'ordre $(1, 0)$ puisque toute chaîne d'ordre $(p > 1, 0)$ a son équivalent d'ordre $(1, 0)$. Les calculs avec ces modèles sont simples et leur représentation peut être réduite à un simple graphe, ce dernier point facilite leur réalisation informatique.

E.6.6 Extension

L'article [Bicego2003] démontre aussi l'équivalence entre un modèle de Markov caché dont les émissions associés aux états sont des mélanges de lois normales avec un modèle de Markov caché dont les émissions sont des lois gaussiennes. Le second modèle comporte bien évidemment plus d'états.

Annexe F

Modèles de Markov cachés et sélection d'architecture

L'objectif de la sélection de l'architecture des modèles de Markov cachés est de trouver le modèle optimal modélisant les séquences d'observations de la base d'apprentissage. Les algorithmes développés dans cette partie ont pour objectif de faire croître et décroître le nombre de coefficients des modèles de Markov cachés jusqu'à ce qu'il se stabilise. La croissance vise à permettre aux modèles de s'adapter à une plus grande variabilité d'images de lettres mais elle génère souvent des coefficients redondants qu'une étape de décroissance tend à supprimer tout en conservant des modèles équivalents ou *proches* (c'est-à-dire dans le cas de la reconnaissance de l'écriture, des modèles dont les performances en reconnaissance sont comparables).

A ce sujet, des travaux ont été menés et ont débouché sur des algorithmes permettant de dire si deux modèles sont équivalents [Ito1992] (coût exponentiel) ou [Balasubramanian1993] (coût polynômial). Mais à ceux-ci seront préférés d'autres algorithmes comme celui développé par [Kamp1985] qui s'intéresse au regroupement de deux états. Ce dernier est en effet adaptable au cas de modèles proches puisque l'équivalence stricte est peu intéressante. En effet, ces modèles sont la solution de problèmes d'optimisation, les méthodes numériques utilisées font fréquemment intervenir une part aléatoire dans la recherche de ces solutions. Celles-ci, pour un même problème, permettent d'obtenir des performances équivalentes bien qu'elles ne soient jamais rigoureusement équivalentes.

Avant même d'apprendre un modèle (de lettre par exemple), la première inconnue est le nombre d'états et les connexions qui les relient entre eux. Cette architecture ne peut être fixée une fois pour toutes car elle dépend des observations, si celles-ci changent (si les graphèmes changent par exemple), la structure de chaque chaîne devra également évoluer. Il faut donc trouver une méthode permettant aux modèles de Markov cachés d'apprendre et de s'adapter.

[Ziv1992] propose un estimateur du nombre d'états mais le calcul de celui-ci suppose l'apprentissage de nombreux modèles, elle n'est pas applicable parce que trop gourmande en calcul et ne prend pas en compte le sens gauche droite de l'écriture. D'autres travaux plus récents adaptent des méthodes utilisées dans d'autres domaines (AIC, BIC, ...) aux cas des modèles de Markov cachés (voir [Durand2003]). Ces méthodes nécessitent toujours l'estimation de nombreux modèles et s'aident parfois d'une évolution guidée de l'architecture (voir [Bicego2003]). Comme dans [Augustin2001], le modèle initial comporte plus de paramètres que le nombre adéquat, auquel sont enlevés états et connexions faibles pour aboutir à une architecture presque équivalente et réduite. Ce chapitre s'inscrit comme complément des méthodes de sélections citées ci-dessus et propose des méthodes de croissance et décroissance de l'architecture des modèles de Markov cachés.

F.1 Propriétés des modèles de Markov cachés

F.1.1 Tests d'adéquation de lois

Le test d'ajustement de Kolmogorov est un test non paramétrique permettant de vérifier si un échantillon $(X_1, \ldots, X_N) \in \mathbb{R}^N$ suit une loi dont la fonction de répartition est $F(x)$ (voir [Saporta1990]). On définit tout d'abord la fonction de répartition empirique $\widehat{F_N}(x)$ par :

$$\widehat{F_N}(x) = \frac{1}{N} \sum_{k=1}^{N} \mathbb{1}_{\{X_k \leqslant x\}} \tag{F.1}$$

On construit la statistique D_N :

$$D_N = \sup_x \left| \widehat{F_N}(x) - F(x) \right| \tag{F.2}$$

Cette statistique est asymptotiquement distribuée selon la loi :

$$\mathbb{P}\left(\sqrt{N}\, D_N < y\right) \overset{N\to\infty}{\longrightarrow} \sum_{-\infty}^{+\infty} (-1)^k \; e^{-2k^2y^2} = K(y) \tag{F.3}$$

La fonction $K(y)$ est tabulée (voir table F.1.1, [Pearson1972]) et permet de définir le test suivant [1] :

$$\left\{ \begin{array}{lcl} H_0 & : & F = F_0 \\ H_1 & : & F \neq F_0 \end{array} \right.$$

n	p = 0.80	p = 0.90	p = 0.95	p = 0.98	p = 0.99
1	0.90000	0.95000	0.97500	0.99000	0.99500
2	0.68377	0.77639	0.84189	0.90000	0.92929
...	...	...	...	...	...
100	0.10563	0.12067	0.13403	0.14987	0.16081
$n > 100$	$\frac{1,073}{\sqrt{n}}$	$\frac{1,223}{\sqrt{n}}$	$\frac{1,358}{\sqrt{n}}$	$\frac{1,518}{\sqrt{n}}$	$\frac{1,629}{\sqrt{n}}$

Tab. F.1: Loi de D_n (F.2), valeur de d_N telle que $\mathbb{P}(D_N < d_n) = p$. Les bases d'images utilisées pour l'estimation des modèles de reconnaissance comportent au moins quelques milliers d'images, donc de grandes valeurs pour n. Seule la dernière ligne sera utilisée.

1. Il existe d'autres tests d'ajustements non paramétriques comme celui de Cramer - von Mises qui s'appuie sur la statistique :

$$N\,\omega_N^2 = \int_{-\infty}^{+\infty} \left[\widehat{F_N}(x) - F_N(x)\right]^2 dF(x) = \frac{1}{12N} + \sum_{k=1}^{N} \left[\frac{2k-1}{2N} - F(X_k)\right]^2 \tag{F.4}$$

La loi de cette statistique est tabulée dans [Pearson1972].

Ce test peut être étendu à la comparaison de deux échantillons indépendants. On dispose donc de deux échantillons $(X_1, \ldots, X_{N_1})$ et $(Y_1, \ldots, Y_{N_2})$ provenant de deux fonctions de répartitions F^1 et F^2 pour lesquels les fonctions de répartition empiriques sont notées $\widehat{F^1_{N_1}}(x)$ et $\widehat{F^2_{N_2}}(x)$. On cherche à résoudre le problème suivant :

$$\left\{ \begin{array}{lcl} H_0 & : & F^1 = F^2 \\ H_1 & : & F^1 \neq F^2 \end{array} \right.$$

Or :

$$\mathbb{P}\left(\sqrt{\frac{N_1 N_2}{N_1 + N_2}} \sup_x \left| \widehat{F^1_{N_1}}(x) - \widehat{F^2_{N_2}}(x) \right| < y \right) \longrightarrow K(y) \tag{F.5}$$

Si on veut contourner la loi tabulée de la table F.1.1, il est possible d'utiliser le test de Wilcoxon-Mann-Whitney pour valider l'hypothèse que deux échantillons $(X_1, \ldots, X_n)$, $(Y_1, \ldots, Y_m)$ suivent la même loi. Pour effectuer de test, on construit la statistique :

$$U = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{1}\!\!1_{\{X_i > Y_j\}} \tag{F.6}$$

Si les deux échantillons suivent la même loi alors $\mathbb{E}\,U = \frac{nm}{2}$ et $\mathbb{V}\,U = \frac{nm(n+m+1)}{12}$. De plus, U tend asymptotiquement vers une loi normale $\mathcal{N}\left(\frac{nm}{2}, \frac{nm(n+m+1)}{12}\right)$.

F.1.2 Etats récurrents, semi-récurrents, cycles

Définition F.1.1 : état semi-récurrent
On appelle un *état semi-récurrent* un état i vérifiant :

$$\forall t \in \mathbb{N},\ \exists\, t' > t \text{ tel que } \mathbb{P}(q_{t'} = i) > 0$$

Autrement dit, si $\min\{d > 0 \mid \forall t \in \mathbb{N},\ \mathbb{P}(q_{t+d} = i \mid q_t = i) > 0\}$ existe et est défini, alors l'état i est un état semi-récurrent.

Les états semi-récurrents sont peu fréquents en reconnaissance de l'écriture car le sens gauche-droite de la lecture implique qu'on ne puisse pas revenir à un état antérieur. Ils introduisent également des incertitudes quant à la longueur de la séquence à reconnaître. Les modèles utilisés jusqu'à présent en reconnaissance de l'écriture n'en possèdent pas et la méthode de sélection d'architecture aura pour objectif de les supprimer.

Définition F.1.2 : état récurrent
On appelle un *état récurrent* un état i vérifiant :

$$\sum_{t=1}^{+\infty} \mathbb{P}(q_t = i \mid q_0 = i) = +\infty$$

Ces états sont inexistants dans les modèles de Markov utilisés lors de la reconnaissance de l'écriture car les séquences d'observations que les modèles doivent apprendre sont finies or un état récurrent implique la possibilité d'émettre des séquences infinies. Les états récurrents sont aussi semi-récurrents.

Définition F.1.3 : cycle
On appelle un *cycle* inclus dans un modèle de Markov un ensemble d'états semi-récurrents (ou récurrents) connectés les uns aux autres. Si un état appartient à ce cycle, il existe une suite de transitions de probabilités non nulles menant de nouveau à cet état.

Si un état fait partie d'un cycle, la longueur de ce cycle est le nombre minimal de transitions nécessaires pour revenir à ce même état. Le plus petit cycle est composé d'un seul état semi-récurrent (ou récurrent).

Partant d'un modèle de Markov caché débarrassé de ses émissions, on désire savoir pour chaque état s'il appartient à un cycle ou non. Pour cela, on définit la probabilité pour un état d'appartenir à un cycle. Le modèle de Markov contient N états numérotés $\{1,\dots,N\}$, une séquence d'états est notée $(q_1,\dots,q_T)$. Soit i un état, $i \in \{1,\dots,N\}$, on définit la probabilité que l'état i soit cyclique :

$$\mathbb{P}\left(i \in cycle\right) = \sum_{t=1}^{+\infty} \mathbb{P}\left(q_t = i \text{ et } q_l \neq i \text{ pour } 0 < l < t \mid q_0 = i\right) = \sum_{t=1}^{+\infty} c_t^i \tag{F.7}$$

La définition de cette probabilité ne permet pas de la calculer explicitement, c'est pourquoi on pose :

$$\begin{aligned} d_t^{ij} &= \mathbb{P}\left(q_t = j \text{ et } q_l \neq i \text{ pour } 0 < l < t \mid q_0 = i\right) \\ c_t^i &= d_t^{ii} \end{aligned}$$

Avec ces notations :

$$\begin{aligned} d_1^{ij} &= \mathbb{P}\left(q_1 = j \mid q_0 = i\right) = a_{ij} \\ d_{t+1}^{ij} &= \mathbb{P}\left(q_{t+1} = j \text{ et } q_l \neq i \text{ pour } 0 < l < t+1 \mid q_0 = i\right) \\ d_{t+1}^{ij} &= \sum_{k \neq i} \mathbb{P}\left(q_{t+1} = j,\, q_t = k \text{ et } q_l \neq i \text{ pour } 0 < l < t \mid q_0 = i\right) \\ d_{t+1}^{ij} &= \sum_{k \neq i} \left(\begin{array}{l} \mathbb{P}\left(q_{t+1} = j \mid q_t = k \text{ et } q_l \neq i \text{ pour } 0 < l < t,\, q_0 = i\right) \\ \mathbb{P}\left(q_t = k \text{ et } q_l \neq i \text{ pour } 0 < l < t \mid q_0 = i\right) \end{array} \right) \end{aligned}$$

D'où :

$$\begin{aligned} d_{t+1}^{ij} &= \sum_{k \neq i} a_{kj}\, d_t^{ik} \\ c_i^t &= d_t^{ii} \end{aligned}$$

Ce cycle a une longueur l_i dont on peut estimer la longueur moyenne L_i :

$$L_i = \mathbb{E}(l_i) = \frac{\sum_{t=1}^{+\infty} t\, c_t^i}{\sum_{t=1}^{+\infty} c_t^i} \tag{F.8}$$

Il peut paraître étonnant d'énoncer certains résultats sur les états semi-récurrents étant donné qu'ils sont peu fréquents en reconnaissance de l'écriture mais ils seront utilisés lors de la sélection automatique de l'architecture.

F.1.3 Distributions temporelles

On considère une chaîne de Markov cachée M incluant un état d'entrée et de sortie. Comme dans le paragraphe précédent, on ne s'intéresse qu'aux états et non aux émissions. On définit $\mathbb{P}(q_t = i)$, la probabilité d'être dans l'état i à l'instant t sachant M (E est l'état d'entrée). On obtient que si N est le nombre d'états du modèle :

$$\mathbb{P}(q_{t+1} = j) = \sum_{i=1}^{N} a_{ij} \mathbb{P}(q_t = i) \tag{F.9}$$

$$\sum_{i=1}^{N} \mathbb{P}(q_1 = i) = 1 \tag{F.10}$$

$$\forall t > 1,\ \sum_{i=1}^{N} \mathbb{P}(q_t = i) = 1 - \sum_{u=1}^{t-1} \underbrace{\mathbb{P}(S \mid u)}_{\substack{\text{probabilité de} \\ \text{sortir à l'instant } u}} \tag{F.11}$$

Où $(\mathbb{P}(q_t = i))_{1 \leqslant i \leqslant N}$ est la distribution des états à l'instant t et $\sum_{u=1}^{t-1} \mathbb{P}(S \mid u)$ est la probabilité d'être sorti du modèle avant l'instant t. L'objectif de ce paragraphe n'est pas d'avoir la distribution des états à chaque instant mais d'estimer la distribution du "temps" pour chaque état :

$$(\mathbb{P}(t \mid i))_{1 \leqslant t < \infty}$$

On s'intéresse d'abord au cas où le temps t est majoré par T. Par conséquent $\mathbb{P}(t) = \frac{1}{T}$; Ce cas correspond à la situation suivante :

$$\forall t > T,\ \forall i,\ \mathbb{P}(q_t = i) = 0$$

On suppose également que :

$$\forall i,\ \sum_{u=1}^{T} \mathbb{P}(q_t = i) > 0$$

Si ce n'est pas le cas pour un état i_0, cet état peut être supprimé du modèle puisqu'il n'intervient jamais. Par conséquent :

$$\mathbb{P}(q_t = i) = \mathbb{P}(i \mid t)$$
$$\mathbb{P}(t \mid i) = \frac{\mathbb{P}(q_t = i)\,\mathbb{P}(t)}{\mathbb{P}(i)} = \frac{\mathbb{P}(q_t = i)\,\mathbb{P}(t)}{\sum_{u=1}^{T} \mathbb{P}(i \mid u)\,\mathbb{P}(u)} = \frac{\frac{1}{T}\mathbb{P}(q_t = i)}{\frac{1}{T}\sum_{u=1}^{T} \mathbb{P}(i \mid u)} = \frac{\mathbb{P}(q_t = i)}{\sum_{u=1}^{T} \mathbb{P}(q_u = i)}$$

Ces résultats sont extensibles au cas où $\sum_{u=1}^{+\infty} \mathbb{P}(q_t = i) < +\infty$. Dans le cas contraire où $\sum_{u=1}^{+\infty} \mathbb{P}(q_t = i) = +\infty$, alors l'état i est *récurrent*. Autrement dit, une fois entré dans l'état i, il n'est plus possible de sortir du modèle, ce qui est impossible dans le cas de la reconnaissance de l'écriture puisque les modèles apprennent des séquences finies d'observations. Les états récurrents sont considérés comme des états "impasses". Si $\sum_{u=1}^{+\infty} \mathbb{P}(q_t = i) < +\infty$, on définit la distribution temporelle de l'état t par :

$$\mathbb{P}(t \mid i) = \frac{\mathbb{P}(q_t = i)}{\sum_{u=1}^{+\infty} \mathbb{P}(q_u = i)} \tag{F.12}$$

Si pour chaque état, on peut définir une distribution temporelle, il est possible également de définir la probabilité d'un état comme étant :

$$\mathbb{P}(i) = \frac{\sum_{u=1}^{+\infty} \mathbb{P}(q_u = i)}{\sum_{i=1}^{N} \sum_{u=1}^{+\infty} \mathbb{P}(q_u = i)} \tag{F.13}$$

Dans le cas général, pour estimer les distributions temporelles des classes d'observations, nous allons construire les matrices suivantes :

N	est le nombre d'états du modèle
T	est une durée
O	est le nombre de classes d'observations
$(\mathbb{P}(i \mid t))_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant t \leqslant T}}$	est la matrice des distributions des états à chaque temps
$(\mathbb{P}(i))_{1 \leqslant i \leqslant N}$	est le vecteur des probabilités des états
$(\mathbb{P}(o \mid i))_{\substack{1 \leqslant i \leqslant N \\ 1 \leqslant o \leqslant O}}$	est la matrice des probabilités d'émissions des observations
$(\mathbb{P}(o))_{1 \leqslant o \leqslant O}$	est le vecteur des probabilités des classes d'observations
$(\mathbb{P}(o \mid t))_{\substack{1 \leqslant o \leqslant O \\ 1 \leqslant t \leqslant T}}$	est la matrice des probabilités des classes d'observations à chaque temps

On en déduit que :

$$\mathbb{P}(o) = \sum_{i=1}^{N} \mathbb{P}(o,i) = \sum_{i=1}^{N} \mathbb{P}(o \mid i)\,\mathbb{P}(i)$$
$$\mathbb{P}(o,t) = \sum_{i=1}^{N} \mathbb{P}(o,i,t) = \sum_{i=1}^{N} \mathbb{P}(o \mid i,t)\,\mathbb{P}(i \mid t)\,\mathbb{P}(t) = \sum_{i=1}^{N} \mathbb{P}(o \mid i)\,\mathbb{P}(i \mid t)\,\mathbb{P}(t)$$

Par conséquent :

$$\begin{array}{rcl} (\mathbb{P}(o)) & = & (\mathbb{P}(o \mid i))'\,(\mathbb{P}(i)) \\ (\mathbb{P}(o \mid t)) & = & (\mathbb{P}(o \mid i))\,(\mathbb{P}(i \mid t))' \end{array} \tag{F.14}$$

F.2 Décroissance de l'architecture

Celle-ci est envisagée en premier parce que sa conception est plus simple. Elle consiste en effet à supprimer les coefficients d'un modèle dont dépendent peu les performances contrairement à la croissance de l'architecture qui tente de combler les carences d'un modèle par l'ajout de coefficients.

F.2.1 Méthode de [Augustin2001]

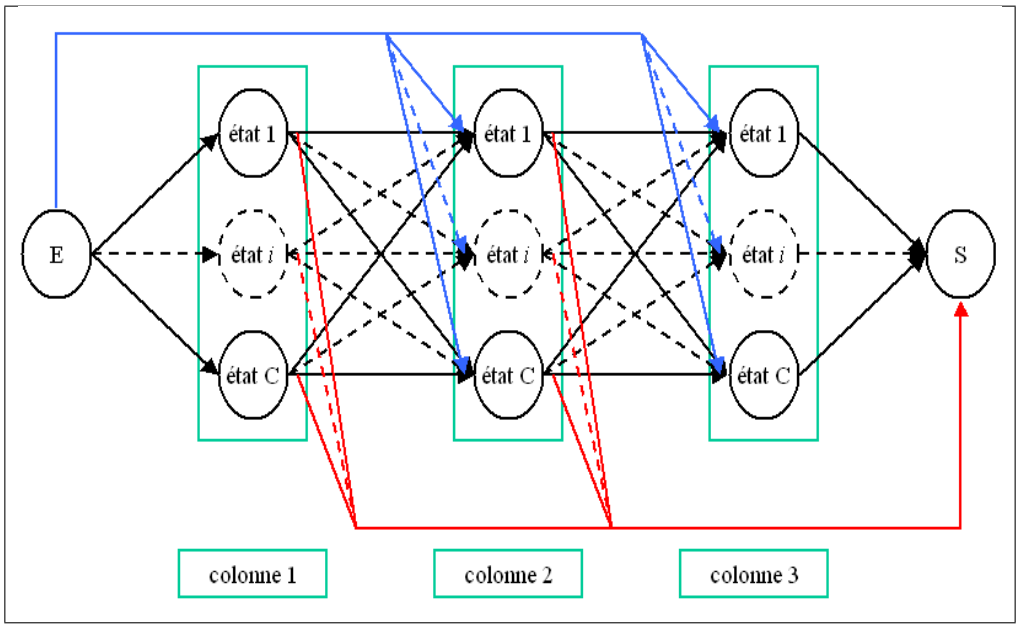


Fig. F.1: Un modèle à structure riche organisé en colonnes (d'après [Augustin2001]). Chaque colonne regroupe ensemble des états susceptibles d'émettre des observations au même instant. Le modèle de la lettre "m", souvent découpée en trois graphèmes, contiendra trois colonnes de plusieurs états, chaque état de la colonne i étant associé à une classe probable pour le $i^{\text{ème}}$ graphème. Cette construction en forme de colonne permet de définir plus facilement une architecture initiale pour le modèle de Markov, soit autant de colonnes qu'il y a de graphèmes. Le nombre d'états par colonne représente la variabilité admise pour chaque graphème.

Les modèles utilisés par [Augustin2001] sont organisés en colonnes d'états (figure F.1), chaque colonne comprend au départ C états où C est le nombre de classes d'observations, chaque état ne peut émettre

qu'une seule classe d'observations (les probabilités d'émission sont dégénérées). La taille de ce modèle riche va décroître, des états et des connexions peu probables vont être suppprimées selon le processus suivant :

Algorithme F.2.1 : décroissance de l'architecture d'après [Augustin2001]

1. Le modèle est appris grâce aux formules de Baum-Welch
2. Pour chaque séquence :
   (a) Le meilleur chemin d'états est obtenu grâce à l'algorithme de Viterbi (ou alignement Viterbi).
   (b) Pour chaque connexion, on compte le nombre de meilleurs chemins l'empruntant.
   (c) On supprime les connexions trop peu utilisées par rapport aux autres (en dessous d'un certain seuil), le nombre de connexions supprimées ne peut être supérieur à un petit nombre ($\sim 10\%$).
   (d) Les états n'étant plus connectés sont supprimés également.
3. On retourne à l'étape 1 tant que l'étape 2 supprime des connexions.

La suppression des connexions s'effectue peu à peu car la suppression de certaines connexions peut amener le modèle à en renforcer d'autres. L'algorithme aboutit à un modèle alléger comme celui par exemple de la figure F.2.

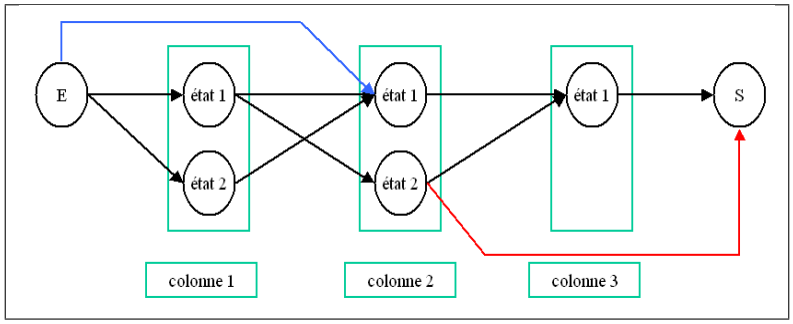


Fig. F.2: Exemple d'architecture sélectionnée par la méthode développée par [Augustin2001].

Le nombre de colonnes maximal que doit contenir le modèle initial correspond au nombre maximal d'observations contenues dans les séquences que le modèle doit apprendre. Par exemple, un modèle associé à la lettre "M" contiendra cinq colonnes tandis que celui associé à la lettre "i" se limitera à deux. Cette méthode fait décroître l'architecture du modèle : états et transitions sont supprimées si la vraisemblance du modèle ne décroît pas beaucoup.

F.2.2 Suppression de connexions peu probables

Soit M une chaîne de Markov cachée comprenant N états et ayant appris les séquences $\left(O^1, \ldots, O^K\right)$ de longueur $(T_1, \ldots, T_K)$. On note S l'ensemble des séquences de ce modèle et L sa vraisemblance.

$$L = \prod_{k=1}^{K} \mathbb{P}\left(O^k | M\right) = \prod_{k=1}^{K} \sum_{s \in S} \mathbb{P}\left(O^k, s | M\right) \tag{F.15}$$

On désire supprimer ou annuler un ensemble C de connexions, on note S' l'ensemble des séquences d'états du modèle utilisant des connexions incluses dans C, M' est le modèle M débarrassé des connexions incluses

dans C. On note L' la vraisemblance du modèle obtenu après suppression des connexions et réestimation des coefficients.

$$L' = \prod_{k=1}^{K} \mathbb{P}\left(O^k | M'\right) = \prod_{k=1}^{K} \sum_{s \in S \setminus S'} \mathbb{P}\left(O^k, s | M'\right)$$

Puisque les coefficients ont été réestimés :

$$L' \geqslant \prod_{k=1}^{K} \left[\sum_{s \in S} \mathbb{P}\left(O^k, s | M\right) - \sum_{s \in S'} \mathbb{P}\left(O^k, s | M\right) \right] \tag{F.16}$$

$$L' \geqslant \prod_{k=1}^{K} \left[\mathbb{P}\left(O^k | M\right) \left[1 - \frac{\sum_{s \in S'} \mathbb{P}\left(O^k, s | M\right)}{\mathbb{P}\left(O^k | M\right)} \right] \right]$$

$$\ln L' \geqslant \sum_{k=1}^{K} \ln \mathbb{P}\left(O^k | M\right) + \sum_{k=1}^{K} \ln \left[1 - \sum_{s \in S'} \mathbb{P}\left(s \left| O^k, M\right.\right) \right]$$

$$\ln L' \geqslant \ln L + \sum_{k=1}^{K} \ln \left[1 - \sum_{s \in S'} \mathbb{P}\left(s \left| O^k, M\right.\right) \right]$$

$$\ln L' - \ln L \geqslant \sum_{k=1}^{K} \ln \left[1 - \sum_{s \in S'} \mathbb{P}\left(s \left| O^k, M\right.\right) \right] \tag{F.17}$$

Pour une séquence d'états donnée s qui passe par la transition c, on sait que $\mathbb{P}\left(s \mid O^k, M\right) \leqslant \mathbb{P}\left(c \mid O^k, M\right)$ puisque $\mathbb{P}\left(c \mid O^k, M\right)$ est la somme des probabilités des séquences d'états incluant la connexion c. On peut donc affirmer que :

$$\sum_{s \in S'} \mathbb{P}\left(s \left| O^k, M\right.\right) \leqslant \sum_{c \in C} \mathbb{P}\left(c \left| O^k, M\right.\right) \tag{F.18}$$

L'expression (F.17) peut à son tour être minorée :

$$\ln L' - \ln L \geqslant \sum_{k=1}^{K} \ln \left[1 - \sum_{c \in C} \mathbb{P}\left(c \left| O^k, M\right.\right) \right] \tag{F.19}$$

$$\ln L' - \ln L \geqslant \sum_{k=1}^{K} \ln \left[1 - \sum_{t=1}^{T_k - 1} \sum_{(i \to j) \in C} \frac{\mathbb{P}\left(q_{t+1} = j, q_t = i, O^k | M\right)}{\mathbb{P}\left(O^k | M\right)} \right] \tag{F.20}$$

En pratique, il est préférable de supprimer plusieurs fois un petit nombre de connexions jusqu'à ce que l'ensemble S' soit vide plutôt que de supprimer les connexions en une seule fois ([Augustin2001]). En effet, dès lors qu'une connexion est supprimée, les probabilités a posteriori réestimées à partir de ce nouveau modèle dont un coefficient est annulé peuvent être très différentes de celles du précédent modèle.

En pratique, comme $\ln(1-x) \sim -x$, on calculera la quantité $\rho_{ij} = \sum_{k=1}^{K} \sum_{t=1}^{T_k - 1} \frac{\mathbb{P}\left(q_{t+1} = j, q_t = i, O^k | M\right)}{\mathbb{P}\left(O^k | M\right)}$. Il faut d'abord classer les connexions selon les ρ_{ij} croissant puis choisir les premières connexions pour construire un ensemble C tel que $\sum_{(i \to j) \in C} \rho_{ij}$ soit petit.

F.2.3 Suppression des états impasses

Lorsque des connexions sont supprimées, certains états peuvent ne plus être atteints ou peuvent ne plus en atteindre d'autres, par exemple : $\forall t > 0, \mathbb{P}(q_t = i) = 0$. Par conséquent, les probabilités $\mathbb{P}(q_{t+1} = j \mid q_t = i)$ n'ont plus de raison d'être et ne peuvent plus être estimées, elles doivent être supprimées également. L'état i est alors un état *impasse*.

Lorsqu'une connexion est supprimée, il faut vérifier qu'aucun état n'est devenu un état impasse, dans le cas contraire, on supprime cet état et les connexions arrivantes et partantes. Ce processus se répète jusqu'à ce que plus aucune connexion ne puisse plus être supprimée.

Le modèle de la figure F.3 possède deux états numérotés 2 et 4 qui sont des états impasse. L'état 2 ne possède aucune connexion entrante excepté celle qui provient de lui-même. L'état 4 ne possède aucune connexion sortante excepté celle qui provient de lui-même. D'autres cas sont moins évidents, le modèle de la figure F.3 un *cycle impasse* : les états 2 et 4 font partie d'un "cycle impasse".

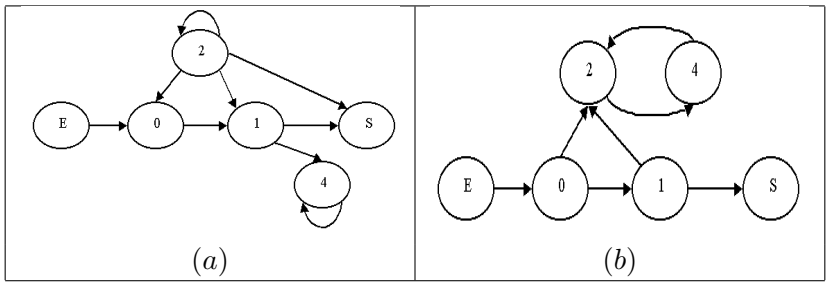


Fig. F.3: Modèle contenant deux états "impasse" (a) et un "cycle impasse" (b).

De ces deux exemples (figure F.3) est inspirée la définition suivante :

> **Définition F.2.2 : état impasse**
> Soit une chaîne de Markov d'ordre 1, un état impasse e est un état qui vérifie l'une des trois conditions suivantes :
>
> 1. il ne possède pas de connexion entrante
> 2. il ne possède pas de connexion sortante
> 3. il est récurrent : $\mathbb{P}(e \in cycle) = 1$

Les états impasse sont supprimés, comme leur suppression peut entraîner la détection de nouveaux états impasse, ce processus est réitéré jusqu'à ce que tous les états de la chaîne de Markov soient valides.

F.2.4 Regroupement d'états *similaires*

F.2.4.1 Principe du regroupement de deux états

Il arrive parfois qu'aucune connexion ne soit négligeable et qu'il soit possible néanmoins d'en supprimer tout en conservant un modèle équivalent. C'est le cas des états similaires, ils jouent le même rôle et les échanger dans le calcul d'une probabilité ne la modifie pas. L'objectif du regroupement de deux états d'un modèle de Markov est donc de réduire le nombre de coefficients tout en conservant un modèle équivalent. Le schéma F.4 illustre le regroupement de deux états en un seul. L'état résultant hérite de toutes les connexions de ses parents.

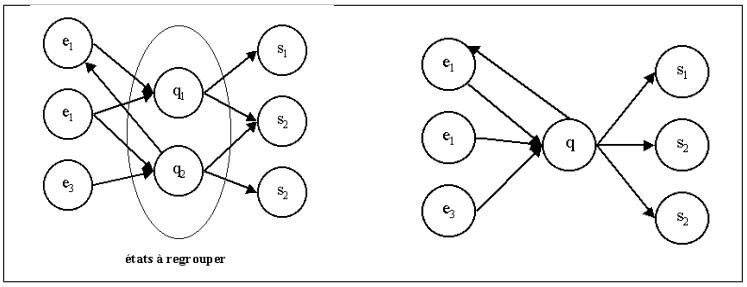


Fig. F.4: Schéma illustrant le regroupement de deux états

On suppose au préalable qu'aucun état du modèle de Markov ne vérifie $\mathbb{P}(i) = \sum_{u=1}^{+\infty} \mathbb{P}(q_t = i) = +\infty$. Le problème consiste à estimer les nouvelles probabilités de transitions de sorte que les deux modèles soient le plus proche possible. Les coefficients du modèle de Markov caché sont les matrices :

$$(\pi, A, \theta, B) \in \mathbb{R}^N \times \mathcal{M}_N(\mathbb{R}) \times \mathbb{R}^N \times \mathcal{M}_{ND}(\mathbb{R})$$

Dans ce modèle, les états x et y vont être regroupés dans un second modèle défini par les matrices :

$$\left(\pi', A', \theta', B'\right) \in \mathbb{R}^{N-1} \times \mathcal{M}_{N-1}(\mathbb{R}) \times \mathbb{R}^{N-1} \times \mathcal{M}_{N-1,D}(R)$$

Les états sont numérotés de 1 à N, et dans le modèle aux deux états regroupés, les états sont numérotés de la même manière à ceci près que l'état numéro y n'existe plus. Par conséquent, le regroupement consiste à regrouper les états x et y dans l'état x :

$$x \longleftarrow x + y$$

E et S sont toujours les états d'entrée et de sortie du modèle. U est l'ensemble $\{E, S\}$, ${}^cU$ est l'ensemble des états émetteurs $(1, \ldots, N)$.

On note $\underset{\longleftrightarrow}{q}$ l'ensemble des prédécesseurs de l'état q, $a_{E,i} = \pi_i$ et $a_{iS} = \theta_i$. L'ensemble des règles qui suivent permettent de calculer les nouvelles transitions et émissions sont les suivantes :

1. Soit $(i,j) \in ({}^cU \cup U)^2$, si $\underset{\longleftrightarrow}{i} \cap (x,y) = \emptyset$ et $j \notin (x,y)$, alors $a'_{ij} = a_{ij}$
2. Soit $i \in ({}^cU \cup U)$, si $i \notin \{x,y\}$, alors $a'_{ix} = a_{ix} + a_{iy}$
3. Soit $i \in ({}^cU \cup U)$, si $i \notin \{x,y\}$, alors $a'_{xi} = \dfrac{\mathbb{P}(x)\, a_{xi} + \mathbb{P}(y)\, a_{yi}}{\mathbb{P}(x) + \mathbb{P}(y)}$ avec $\mathbb{P}(x)$ étant la probabilité de l'état x définie dans le paragraphe F.1.3.
4. $a'_{xx} = \dfrac{\mathbb{P}(x)(a_{xx} + a_{xy}) + \mathbb{P}(y)(a_{yx} + a_{yy})}{\mathbb{P}(x) + \mathbb{P}(y)}$
5. Soit $i \in ({}^cU \cup U)$, si $i \notin \{x,y\}$, alors $\forall o,\ b'_i(o) = b_i(o)$
6. $\forall o,\ b'_{xo} = \dfrac{\mathbb{P}(x)\, b_{xo} + \mathbb{P}(y)\, b_{yo}}{\mathbb{P}(x) + \mathbb{P}(y)}$

On vérifie bien que :

$$\forall i \in ({}^cU \cup U), \sum_{j \in ({}^cU \cup U)} a'_{ij} = 1$$

Les règles 3,4,6 se déduisent du fait que l'état regroupé x' est la réunion des deux états x et y : $x' = x \cup y$. Par conséquent, en appliquant la règle de Bayes, on trouve :

$$\begin{aligned}\mathbb{P}\left(q_{t+1} = i \mid q_t = j \neq x',\, i = x'\right) &= \mathbb{P}\left(q_{t+1} = i \mid q_t = j \neq x',\, i = x\right)\mathbb{P}\left(i = x\right) + \\ &\quad \mathbb{P}\left(q_{t+1} = i \mid q_t = j \neq x',\, i = y\right)\mathbb{P}\left(i = y\right)\end{aligned}$$

$$\mathbb{P}\left(q_{t+1} = i \mid q_t = j \neq x',\, i = x'\right) = \frac{\mathbb{P}\left(q_{t+1} = x \mid q_t = j \neq x'\right)\mathbb{P}\left(x\right) + \mathbb{P}\left(q_{t+1} = y \mid q_t = j \neq x'\right)\mathbb{P}\left(y\right)}{\mathbb{P}\left(x\right) + \mathbb{P}\left(y\right)}$$

$$\mathbb{P}\left(q_{t+1} = x' \mid q_t = x'\right) = \frac{\left[\begin{array}{ll}\left[\mathbb{P}\left(q_{t+1} = x \mid q_t = x\right) + \mathbb{P}\left(q_{t+1} = y \mid q_t = x\right)\right]\mathbb{P}\left(x\right) & + \\ \left[\mathbb{P}\left(q_{t+1} = x \mid q_t = y\right) + \mathbb{P}\left(q_{t+1} = y \mid q_t = y\right)\right]\mathbb{P}\left(y\right) & \end{array}\right]}{\mathbb{P}\left(x\right) + \mathbb{P}\left(y\right)}$$

Il en est de même pour la règle 6.

F.2.4.2 Premier théorème

Le principe du regroupement exposé ci-dessus est valable quelle que soit la paire d'états choisie dans le modèle de Markov caché, néanmoins, tous les regroupements ne mènent pas à un modèle équivalent.

Théorème F.2.3 : regroupement d'états
Soit M un modèle de Markov à N états, soit $(i,j) \in (1,\ldots,N)^2$ et $i \neq j$, on appelle M'_{ij} le modèle où les états i et j sont regroupés dans l'état i selon les règles 1 à 6 exposées ci-dessus (paragraphe F.2.4.1). Si les deux conditions suivantes sont vérifiées :

$$\left\{\begin{array}{l} 1 - \text{les distributions d'émissions des états } i \text{ et } j \text{ sont égales} \\ 2 - \forall\,(k,l) \in (E,S,1,...,N)^2\,,\ \forall d \geqslant 1, \\ \qquad \mathbb{P}\left(q_{t+d+1} = l,\ (\forall t' \in \{1,\ldots,d\}\,,\ q_{t+t'} \in \{i,j\})\ \mid q_t = k,\ M\right) = \\ \qquad\quad \mathbb{P}\left(q_{t+d+1} = l,\ (\forall t' \in \{1,\ldots,d\}\,,\ q_{t+t'} = i)\ \mid q_t = k,\ M'_{ij}\right) \end{array}\right. \tag{F.21}$$

alors M est équivalent à M'_{ij}

Démonstration (théorème F.2.3) :

Soit $O \in \mathcal{O}$ une séquence d'observations. $O = (O_1,\ldots,O_T) \in \mathcal{O}$. Soit $S = (E, q_1, ..., q_T, S) = (q_0, q_1, ..., q_T, q_{T+1})$ une séquence d'états. Tout d'abord, comme les deux états ont la même distribution de probabilités d'émissions, les règles de regroupement des états impliquent que :

$$\mathbb{P}\left(O_t \mid q_t = i, M\right) = \mathbb{P}\left(O_t \mid q_t = j, M\right) = \mathbb{P}\left(O_t \mid q_t = i, M'_{ij}\right) = b_i\left(O_t\right)$$

On définit la suite $(t_u)_{0 \leqslant u \leqslant U}$ par :

$$\left\{\begin{array}{l} \forall u,\ t_u < t_{u+1} \\ \forall u,\ q_{t_u} \notin \{i,j\} \\ \forall t \in (0,...,T+1)\,,\ q_t \notin \{i,j\} \Longrightarrow t \in (t_0,...,t_U) \end{array}\right.$$

Cette suite représente simplement les indices temporels croissants des états qui ne sont ni i ni j. On note également :

$$\begin{cases} \left(A_{kl}^{d}\right)^{M} &= \mathbb{P}\left(q_{t+d+1}=l,\, (\forall t' \in (1,\ldots,d)\,,\, q_{t+t'} \in \{i,j\}) \mid q_t=k,\, M\right) \\ \left(A_{kl}^{d}\right)^{M'_{ij}} &= \mathbb{P}\left(q_{t+d+1}=l,\, (\forall t' \in (1,\ldots,d)\,,\, q_{t+t'}=i) \mid q_t=k,\, M'_{ij}\right) \end{cases}$$

La probabilité d'une séquence est :

$$\mathbb{P}\left(O \mid M\right) = \sum_{(q_1,\ldots,q_T)} \mathbb{P}\left(O \mid (E, q_1, ..., q_T, S)\,,\, M\right)$$

On en déduit que :

$$\begin{aligned} \mathbb{P}\left(O \mid S, M\right) &= a_{q_T,q_T} \prod_{t=1}^{T+1} a_{q_{t-1},q_t}\, b_{q_t}\left(O_t\right) \\ \mathbb{P}\left(O \mid S, M\right) &= \prod_{u=1}^{U} \left[\begin{array}{l} \mathbb{1}_{\{t_{u-1}+1=t_u\}} a_{q_{t_{u-1}},q_{t_u}}\, b_{q_{t_u}}\left(O_t\right) + \\ \mathbb{1}_{\{t_{u-1}+1\neq t_u\}}\, \left(A_{kl}^{d}\right)^{M} \left[b_i\left(O_t\right)\right]^{\left(t_u - t_{u-1} - \mathbb{1}_{\{t_u=T+1\}}\right)} \end{array} \right] \\ \mathbb{P}\left(O \mid S, M\right) &= \prod_{u=1}^{U} \left[\begin{array}{l} \mathbb{1}_{\{t_{u-1}+1=t_u\}} a_{q_{t_{u-1}},q_{t_u}}\, b_{q_{t_u}}\left(O_t\right) + \\ \mathbb{1}_{\{t_{u-1}+1\neq t_u\}}\, \left(A_{kl}^{d}\right)^{M'_{ij}} \left[b_i\left(O_t\right)\right]^{\left(t_u - t_{u-1} - \mathbb{1}_{\{t_u=T+1\}}\right)} \end{array} \right] \\ \mathbb{P}\left(O \mid S, M\right) &= \mathbb{P}\left(O \mid S, M'_{ij}\right) \end{aligned}$$

Le théorème est démontré.

(F.2.3) □

F.2.4.3 Exemple

Le cas figure F.5 illustre un cas pour lequel la condition 2 est vérifiée. On veut regrouper les états 2 et 3 qui ont tous deux la même probabilité d'émission. Les probabilités de transition nécessaires au théorème F.2.3 sont :

$$\begin{cases} \left(A_{12}^{1}\right)^{M} = a \\ \left(A_{15}^{1}\right)^{M} = b \\ \left(A_{16}^{1}\right)^{M} = a+b \\ \forall k,\, d>1 \Longrightarrow \left(A_{1k}^{d}\right)^{M} = 0 \end{cases}$$

Le schéma F.6 montre le modèle équivalent à celui de la figure F.5 dans lequel les états 1 et 2 ont été regroupés. D'après les six règles énoncés au paragraphe F.2.4.1 :

$$\begin{aligned} c &= a'_{12} = a_{12} + a_{13} = a+b \\ d &= a'_{24} = \frac{\mathbb{P}\left(1\right) a_{24} + \mathbb{P}\left(3\right) a_{34}}{\mathbb{P}\left(2\right) + \mathbb{P}\left(3\right)} = \frac{\mathbb{P}\left(2\right) a_{24} + \mathbb{P}\left(3\right) a_{34}}{\mathbb{P}\left(2\right) + \mathbb{P}\left(3\right)} = \frac{\mathbb{P}\left(2\right)}{\mathbb{P}\left(2\right) + \mathbb{P}\left(3\right)} = \frac{a}{a+b} \end{aligned}$$

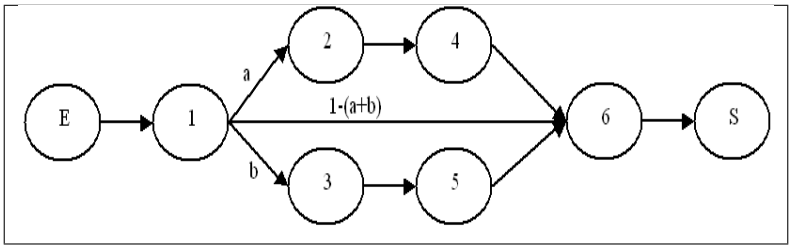


Fig. F.5: Exemple où le théorème F.2.3 s'applique.

$tat \rightarrow t$	1	2	3	4	5	*somme*	$P(.)$
1	1	0	0	0	0	1	$\frac{1}{2(1+a+b)}$
2	0	a	0	0	0	a	$\frac{a}{2(1+a+b)}$
3	0	b	0	0	0	b	$\frac{b}{2(1+a+b)}$
4	0	0	a	0	0	a	$\frac{a}{2(1+a+b)}$
5	0	0	b	0	0	b	$\frac{b}{2(1+a+b)}$
6	0	$1-a+b$	0	$a+b$	0	1	$\frac{1}{2(1+a+b)}$

Tab. F.2: Distribution temporelle des états pour le modèle figure F.5.

On en déduit que :

$$\begin{cases} \left(A_{14}^{1}\right)^{M'_{23}} = cd = (a+b)\,\dfrac{a}{a+b} = a = \left(A_{14}^{1}\right)^{M} \\ \left(A_{15}^{1}\right)^{M'_{23}} = c\,(1-d) = (a+b)\left(1-\dfrac{a}{a+b}\right) = b = \left(A_{15}^{1}\right)^{M} \\ \left(A_{16}^{1}\right)^{M'_{23}} = 1-c = 1-a-b = \left(A_{16}^{1}\right)^{M} \\ \forall k,\, d>1 \Longrightarrow \left(A_{1k}^{d}\right)^{M'_{23}} = 0 \end{cases}$$

Les deux modèles sont bien équivalents.

F.2.4.4 Second théorème

Lorsqu'on veut regrouper les états i et j, le théorème précédent possède un inconvénient car il nécessite de calculer les probabilités de transiter d'un état à un autre quel que soit le nombre d'états intercalés entre i

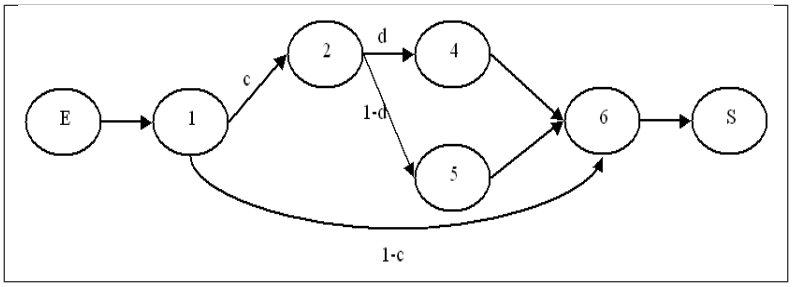


Fig. F.6: Modèle résultant du regroupement des états 2 et 3 dans le modèle figure F.5.

et j. Le suivant (voir aussi [Kamp1985]) se propose d'affaiblir cette condition au profit d'une autre :

Théorème F.2.4 : regroupement d'états d'après [Kamp1985]
Soit M un modèle de Markov à N états, soit $(i,j) \in (1,\ldots,N)^2$ et $i \neq j$. On appelle M'_{ij} le modèle où les états i et j sont regroupés dans l'état i selon les règles 1 à 6 exposées ci-dessus (paragraphe F.2.4.1). Si les trois conditions suivantes sont vérifiées :

$$\begin{cases} 1- & \text{les distributions d'émissions des états } i \text{ et } j \text{ sont égales} \\ 2- & \forall\,(k,l) \in (E,S,1,...,N)^2\,, \\ & \qquad \mathbb{P}\left(q_{t+2}=l,\, q_{t+1} \in (i,j) \mid q_t = k,\, M\right) = \mathbb{P}\left(q_{t+2}=l,\, q_{t+1}=i \mid q_t=k,\, M'_{ij}\right) \\ 3- & \forall t>0, \forall k \in (1,\ldots,N)\,,\; k \notin (i,j) \Longrightarrow \mathbb{P}\left(k \mid t, M\right) = \mathbb{P}\left(k \mid t, M'_{ij}\right) \\ & \forall t>0,\; \mathbb{P}\left(i \mid t, M\right) + \mathbb{P}\left(j \mid t, M\right) = \mathbb{P}\left(i \mid t, M'_{ij}\right) \end{cases}$$

alors M est équivalent à M'_{ij}.

Démonstration (théorème F.2.4) :
Il est évident que la condition 2 du théorème 1 implique les conditions 2 et 3 du thèorème 2. Mais ici, il s'agit de démontrer la réciproque :

$$\begin{cases} \forall\,(k,l) \in \{E,S,1,...,N\}^2 \\ \qquad \mathbb{P}\left(q_{t+2}=l,\, q_{t+1} \in \{i,j\} \mid q_t=k,\, M\right) = \mathbb{P}\left(q_{t+2}=l,\, q_{t+1}=i \mid q_t=k,\, M'_{ij}\right) \\ \forall t>0, \forall k \in \{1,\ldots,N\}\,,\; k \notin \{i,j\} \Longrightarrow \mathbb{P}\left(k \mid t, M\right) = \mathbb{P}\left(k \mid t, M'_{ij}\right) \\ \forall t>0,\; \mathbb{P}\left(i \mid t, M\right) + \mathbb{P}\left(j \mid t, M\right) = \mathbb{P}\left(i \mid t, M'_{ij}\right) \end{cases}$$

$$\Longrightarrow \quad \begin{cases} \forall\,(k,l) \in (E,S,1,...,N)^2\,,\; \forall d \geqslant 1, \\ \qquad \mathbb{P}\left(q_{t+d+1}=l,\, (\forall t' \in \{1,\ldots,d\}\; q_{t+t'} \in \{i,j\}) \mid q_t=k,\, M\right) = \\ \qquad\quad \mathbb{P}\left(q_{t+d+1}=l,\, (\forall t' \in \{1,\ldots,d\}\; q_{t+t'}=i) \mid q_t=k,\, M'_{ij}\right) \end{cases}$$

On sait que $\mathbb{P}\left(l \mid t=1\right) = \pi_l$. Or par récurrence, on obtient que :

$$\forall l \in \{1,\ldots,N\}\,,\, \mathbb{P}\left(l \mid t+1\right) = \sum_{k=1}^{N} a_{kl} \mathbb{P}\left(k \mid t\right)$$

Par conséquent :

$$
\begin{aligned}
\mathbb{P}\left(l \mid t=1, M\right) &= \pi_l \\
\forall l \notin \{i,j\}\,,\ \mathbb{P}\left(l \mid t=2, M\right) &= \sum_{k=1}^{N} \pi_k\, a_{kl} = \left[\sum_{k \notin \{i,j\}} \pi_k a_{kl}\right] + \pi_i a_{il} + \pi_j a_{jl} \\
\forall n \notin \{i,j\}\,, \quad \mathbb{P}\left(n \mid t=3, M\right) &= \left[\sum_{(k,l) \notin \{i,j\}^2} \pi_k a_{kl} a_{l,n}\right] + \pi_i a_{ii} a_{in} + \pi_i a_{ij} a_{jn} + \pi_j a_{ji} a_{in} + \pi_j a_{jj} a_{jn} \\
\text{or } \forall n \notin \{i,j\}\,, \quad \mathbb{P}\left(n \mid t=3, M'_{ij}\right) &= \left[\sum_{(k,l) \notin \{i,j\}^2} \pi'_k a'_{kl} a'_{l,n}\right] + \pi'_i a'_{ii} a'_{in}
\end{aligned}
$$

D'après les hypothèses :

$$
\begin{aligned}
&\forall n \notin \{i,j\}\,, \\
&\qquad \mathbb{P}\left(n \mid t=3, M'_{ij}\right) - \mathbb{P}\left(n \mid t=3, M\right) = 0 \\
&\qquad \Longrightarrow \pi'_i\, a'_{ii}\, a'_{in} - \left(\pi_i\, a_{ii}\, a_{in} + \pi_i\, a_{ij}\, a_{jn} + \pi_j a_{ji}\, a_{in} + \pi_j\, a_{jj}\, a_{jn}\right) = 0 \\
&\qquad \Longrightarrow \left[\begin{array}{ll} & \mathbb{P}\left(q_{t+3}=n,\, (q_{t+2}, q_{t+1}) \in \{i,j\}^2 \mid q_t = E,\, M\right) \\ - & \mathbb{P}\left(q_{t+3}=n,\, q_{t+2}=i,\, q_{t+1}=i \mid q_t = E,\, M'_{ij}\right) \end{array}\right] = 0
\end{aligned}
$$

De même, pour $t=4$, on obtient que :

$$
\begin{aligned}
&\forall m \notin \{i,j\}\,, \forall n \notin \{i,j\}\,, \\
&\qquad \mathbb{P}\left(n \mid t=4, M'_{ij}\right) - \mathbb{P}\left(n \mid t=4, M\right) = 0 \\
&\qquad \Longrightarrow a'_{mi}\, a'_{ii}\, a'_{in} - \left(a_{mi}\, a_{ii}\, a_{in} + a_{mi}\, a_{ij}\, a_{nm} + a_{mj}\, a_{ji}\, a_{in} + a_{mi}\, a_{jj}\, a_{jn}\right) = 0 \\
&\qquad \Longrightarrow \left[\begin{array}{ll} & P\left(q_{t+3}=n,\, (q_{t+2}, q_{t+1}) \in \{i,j\}^2 \mid q_t = m,\, M\right) \\ - & P\left(q_{t+3}=n,\, q_{t+2}=i,\, q_{t+1}=i \,\middle|\, q_t = m,\, M'_{ij}\right) \end{array}\right] = 0
\end{aligned}
$$

Par récurrence sur t, on démontre que :

$$
\left\{\begin{array}{l} \forall\, (k,l) \in \{E, S, 1, ..., N\}^2\,,\ \forall d \geqslant 1, \\ \qquad P\left(q_{t+d+1} = l,\, (\forall t' \in \{1, ..., d\}\ q_{t+t'} \in \{i,j\}) \mid q_t = k,\, M\right) = \\ \qquad\quad P\left(q_{t+d+1} = l,\, (\forall t' \in \{1, ..., d\}\ q_{t+t'} = i) \,\middle|\, q_t = k,\, M'_{ij}\right) \end{array}\right.
$$

Le théorème est démontré.

(F.2.4) □

Remarque F.2.5: différence
Ce théorème n'est pas rigoureusement identique à celui présenté dans [Kamp1985] car les hypothèses sur les états sont différentes. [Kamp1985] suppose que les états i et j devant être regroupés ne sont pas liés :

$$
\forall d > 0,\ \mathbb{P}\left(q_{t+d} = j \mid q_t = i\right) = \mathbb{P}\left(q_{t+d} = j \mid q_t = i\right) = 0
$$

Avec cette restriction, les hypothèses 1 et 2 du thèorème F.2.4 impliquent l'hypothèse 3.

Remarque F.2.6: associativité du regroupement d'états
Etant donné l'interchangeabilité de deux états similaires, trois états similaires deux à deux peuvent être regroupés en un seul état, d'abord par deux, puis ce nouvel état avec le dernier, et ce, dans n'importe quel ordre.

Remarque F.2.7: équivalence
Les deux théorèmes (F.2.3 et F.2.4) supposent que les deux états ont été regroupés selon les règles 1 à 6 (paragraphe F.2.4.1) et s'intéressent à l'équivalence du modèle M' aux états regroupés avec le modèle initial M. Si le modèle M correspond au maximum de la vraisemblance des observations, alors le modèle M' aussi. Si tel n'était pas le cas, alors il serait possible à partir du modèle M' de construire un modèle ayant la même architecture que le modèle M mais garantissant une meilleure vraisemblance.

F.2.4.5 Mise en pratique

Le paragraphe F.2.4 a introduit la notion d'équivalence entre modèles. La définition de cette notion est trop stricte et ne permet pas de faire évoluer convenablement les modèles : une notion plus souple de "similarité" va être introduite. Jusqu'à présent, nous avons vu comment augmenter le nombre d'états et réduire le nombre de transitions, les paragraphes suivant montrent comment réduire le nombre d'états en procédent par regroupements successifs d'états "similaires".

Définition F.2.8 : modèles similaires
Soient deux modèles de Markov cachés M_1 et M_2, soit $\mathbf{O} = \left(O^k\right)_{1 \leqslant k \leqslant K}$ une suite de séquences d'observations. M_1 et M_2 sont "similaires" ou "presque équivalents" pour $\left(O^k\right))_{1 \leqslant k \leqslant K}$ si le test suivant est validé :

$$H_0 : \text{les distributions } \left(\mathbb{P}\left(o \mid M_1\right)\right)_{o \in \mathbf{O}} \text{ et } \left(\mathbb{P}\left(o \mid M_2\right)\right)_{o \in \mathbf{O}} \text{ sont égales}$$
$$H_1 : \text{les distributions } \left(\mathbb{P}\left(o \mid M_1\right)\right)_{o \in \mathbf{O}} \text{ et } \left(\mathbb{P}\left(o \mid M_2\right)\right)_{o \in \mathbf{O}} \text{ sont différentes}$$

Définition F.2.9 : états similaires
Soit un modèle de Markov caché M et (i, j) deux de ses états, soit $\mathbf{O} = \left(O^k\right)_{1 \leqslant k \leqslant K}$ une suite de séquences d'observations. (i, j) sont des états "similaires" ou "presque équivalents" pour $\left(O^k\right)_{1 \leqslant k \leqslant K}$ si le modèle M'_{ij} résultant du regroupement de i et j est similaire ou presque équivalent à M pour $\left(O^k\right)_{1 \leqslant k \leqslant K}$.

Avant de regrouper deux états, il convient de les choisir et le meilleure moyen de vérifier que deux modèles sont presque équivalents est de calculer la vraisemblance obtenue pour ces deux modèles sur la base de séquences d'apprentissage $\left(O^1, \ldots, O^K\right)$. Dans le cas de grandes bases de données, il est impossible d'effectuer ce calcul pour chaque regroupement d'états possible, donc pour chaque couple d'états. Les paragraphes qui suivent proposent une méthode de sélection de ce couple à partir de calcul effectués sur le modèle initial, les propriétés décrites au paragraphe F.1 seront largement utilisées. Pour commencer,

nous allons supposer que deux états sont presque similaires s'ils vérifient les quatre règles suivantes :

Définition F.2.10 : quatre règles pour détecter les états similaires

Règle 1 : Les deux distributions de leur probabilités d'émissions $(\mathbb{P}(o \mid x))_{1 \leqslant o \leqslant D}$ et $(\mathbb{P}(o \mid y))_{1 \leqslant o \leqslant D}$ pour chacun de ces états sont similaires : l'hypothèse que les deux distributions sont égales est validée par un test de Wilcoxon-Mann-Whitney (F.6).

Règle 2 : La distribution jointe temporelle des deux états x et y est similaire à la distribution temporelle de l'état résultant du regroupement de ces deux états : l'hypothèse que la distribution de l'état regroupant x et y est la même que la distribution jointe des deux états x et y est validée par un test de Kolmogorov (F.2).

Règle 3 : Les distributions temporelles des autres états sont peu modifiées par le regroupement des deux états x et y : pour chaque état différent de x et y, l'hypothèse que la distribution de l'état dans le modèle où x et y ont été regroupés est la même que la distribution du même état dans le modèle de départ est validée par un test de Kolmogorov (F.2).

Règle 4 : Pour tout k, les distributions

$$\mathbb{P}(q_{t+2} = l,\, q_{t+1} \in \{x, y\} \mid q_t = k,\, M)_l \text{ et } \mathbb{P}\left(q_{t+2} = l,\, q_{t+1} = x \mid q_t = k,\, M'_{xy}\right)_l$$

sont similaires, pour chaque état $k \notin \{i, j\}$, l'hypothèse que la distribution

$$\mathbb{P}(q_{t+2} = l,\, q_{t+1} \in \{x, y\} \mid q_t = k,\, M)_l$$

est la même que :

$$\mathbb{P}\left(q_{t+2} = l,\, q_{t+1} = x \mid q_t = k,\, M'_{xy}\right)_l$$

est validée par un test Kolmogorov (F.2).

A chacune de ces règles correspond un test décrit dans les paragraphes suivants. Or pour appliquer ce test, il faut tenir compte de la base d'apprentissage sur laquelle ont été apprises les transitions. Il faut donc mesurer la taille des échantillons qui ont servi à estimer les distributions de probabilités nécessaires au test. Pour une séquence O^k, $T\left(O^k\right)$ est la longueur de cette séquence, pour le modèle M, nous pouvons écrire :

$$\mathbb{P}\left(q_t = i, O^k \mid t\right) = \alpha_t^k(i)\ \beta_t^k(i) \Longrightarrow \mathbb{P}\left(i, O^k\right) = \frac{1}{T\left(O^k\right)} \sum_{t=1}^{T\left(O^k\right)} \alpha_t^k(i)\ \beta_t^k(i)$$

Par conséquent, on peut estimer la probabilité d'un état pour l'ensemble de la base d'apprentissage :

$$P_{app}(i) = \sum_{k=1}^{K} \frac{1}{T\left(O^k\right)} \sum_{t=1}^{T\left(O^k\right)} \alpha_t(i)\ \beta_t(i) \tag{F.22}$$

Nous allons supposer que les quatre tests correspondant aux quatre règles sont indépendants.

Test F.2.11 : Test associé à la règle 1
On note :

$$p_o^{xy} = \frac{P_{app}(x)\ \mathbb{P}(o \mid x) + P_{app}(y)\,\mathbb{P}(o \mid y)}{P_{app}(x) + P_{app}(y)}$$

On applique deux fois le test de Wilcoxon-Mann-Whitney entre les deux distributions $\left(p_o^{xy}\right)_o$ et $\left(\mathbb{P}(o \mid y)\right)_o$ et entre les deux distributions $\left(p_o^{xy}\right)_o$ et $\left(\mathbb{P}(o \mid x)\right)_o$.

Test F.2.12 : Test associé à la règle 2
Les distributions temporelles des états x et y sont :

$$\left(\mathbb{P}(t \mid x)\right)_{0\leqslant t<+\infty} \text{ et } \left(\mathbb{P}(t \mid y)\right)_{0\leqslant t<+\infty}$$

La distribution temporelle de l'état résultat de leur regroupement est :

$$\left(\mathbb{P}\left(t \mid \binom{x}{y}\right)\right)_{0\leqslant t<+\infty}$$

On note également : $P_{app}\binom{x}{y} = P_{app}(x) + P_{app}(y)$, on note :

$$q_t^{xy} = \frac{P_{app}(X)\,\mathbb{P}(t \mid x) + P_{app}(Y)\,\mathbb{P}(t \mid y)}{P_{app}(X) + P_{app}(Y)}$$

On applique le test de Kolmogorov aux deux distributions $\left(\mathbb{P}\left(t \mid \binom{x}{y}\right)\right)_t$ et $\left(q_t^{xy}\right)_t$.

Test F.2.13 : Test associé à la règle 3
La distribution temporelle d'un état $z \notin \{x, y\}$ est $\left(\mathbb{P}(t \mid z)\right)_{0\leqslant t<+\infty}$. La distribution temporelle du même état dans le modèle dans lequel les états x et y ont été regroupés est :

$$\left(\mathbb{P}\left(t \mid z, \binom{x}{y}\right)\right)_{0\leqslant t<+\infty}$$

On applique le test de Kolmogorov aux deux distributions $\left(\mathbb{P}\left(t \mid z, \binom{x}{y}\right)\right)_t$ et $\left(\mathbb{P}(t \mid z)\right)_t$.

Test F.2.14 : Test associé à la règle 4
Soit un état $k \notin \{x, y\}$ est $\left(\mathbb{P}(t \mid z)\right)_{0\leqslant t<+\infty}$. On pose :

$$\forall l \in \{1, ..., N\},\ q_l = \mathbb{P}\left(q_{t+2} = l \mid q_t = k,\ q_{t+1} \in \{x, y\},\ M\right)$$
$$\forall l \in \{1, ..., N\},\ p_l = \mathbb{P}\left(q_{t+2} = l \mid q_t = k,\ q_{t+1} = x,\ M'_{xy}\right)$$

On applique le test de Kolmogorov aux deux distributions $(q_l)_l$ et $(p_l)_t$.

Test F.2.15 : Test associé aux quatre règles
Si les quatre tests F.2.11 à F.2.14 sont validés alors l'hypothèse que les états x et y sont similaires

Ce dernier test permet de savoir si deux états sont similaires ou non et qui par conséquent doivent être regroupés. Il reste à déterminer dans quel ordre les paires d'états similaires doivent être regroupées. Il est possible de classer ces paires d'états par ordre de statistique croissant (que ce soit celle de Wilcoxon-Mann-Whitney ou celle de Kolmogorov). La meilleure paire d'états est celle qui obtient les rangs les plus faibles pour chacun des tests. Comme pour la méthode de suppression des connexions, on regroupe des paires d'états tant qu'au moins une d'entre elles vérifie le test F.2.15 et on évite de regrouper trop de paires sans réestimer les paramètres du modèle entre temps.

F.3 Croissance de l'architecture

Les méthodes décroissantes se soucient peu du problème modélisé à l'inverse des méthodes croissantes, celles-ci cherchent à améliorer les performances des modèles en augmentant le nombre de coefficients qu'ils contiennent. Parmi les trois méthodes proposées, deux sont restreintes au problème de la reconnaissance de l'écriture manuscrite. L'écriture s'effectue de gauche à droite ce qui impose à tout état déjà visité de ne pas pouvoir l'être à nouveau (pas de cycle). De plus, afin de pouvoir utiliser efficacement des algorithmes de recherches du meilleur chemin, à chaque état d'une chaîne de Markov associée à une lettre devra correspondre une, voire quelques classes du système de classification.

En alternant décroissant et croissance de l'architecture des modèles (voir figure F.7), on espère que le modèle résultant de cette sélection d'architecture est optimal : il n'existe pas d'autres modèles augmentant de manière significative la vraisemblance des observations et à vraisemblance constante, le modèle obtenu est le plus petit.

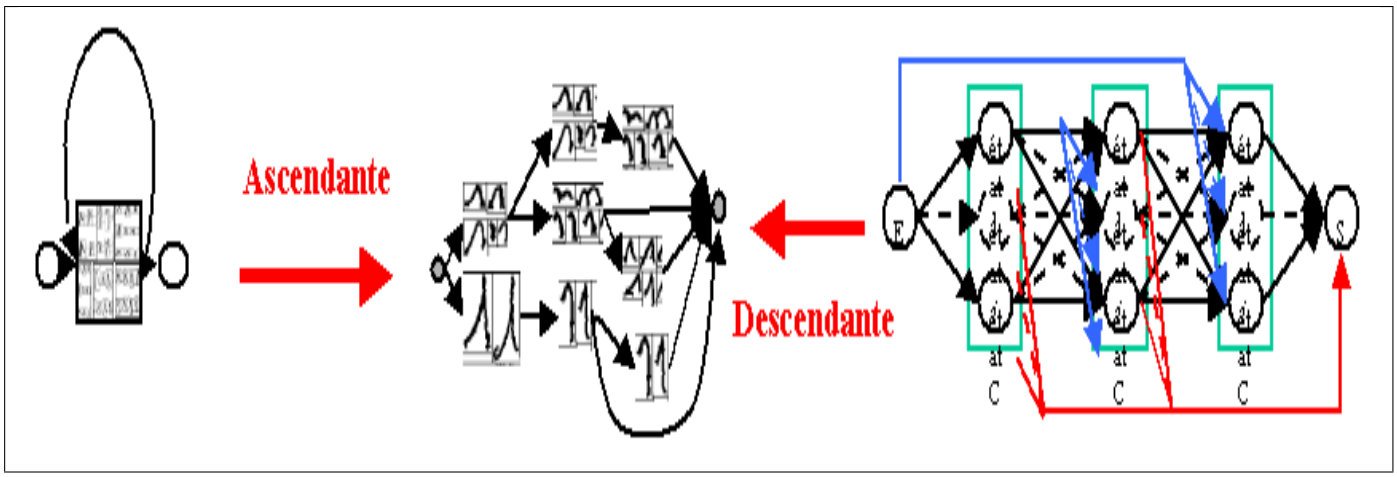


Fig. F.7: Sélection de l'architecture : croissance et décroissance.

F.3.1 Multiplication d'états

L'architecture d'un modèle de Markov peut croître en augmentant le nombre de transitions non nulles ou augmenter le nombre d'états mais cette évolution doit conserver un modèle presque équivalent. Pour chaque architecture A, il est possible de définir L_A la vraisemblance maximum pour cette structure. Lorsque A augmente pour A', alors la vraisemblance doit vérifier : $L_{A'} \geqslant L_A$. Cependant, l'apprentissage de ces grands modèles peut parfois échouer à cause de nombreux maxima locaux pour lesquels $L_{A'} < L_A$. L'idée de la multiplication des états consiste à créer une architecture A' équivalente à A, donc de même

vraisemblance mais incluant plus de coefficients. De cette manière, la croissance de la vraisemblance est toujours assurée.

La multiplication d'un état q consiste à ajouter au modèle d'autres états qui sont de parfaites copies de l'original : remplacer l'état original par une de ses copies dans une séquence d'états ne change pas la probabilité conjointe de cette séquence d'états et celle des observations. La figure F.8 représente l'ensemble des connexions et états impliqués dans la multiplication de l'état q.

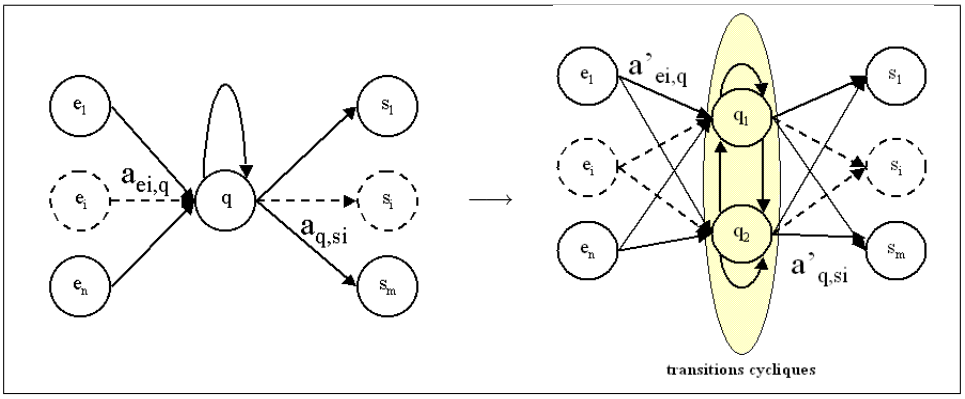


Fig. F.8: Schéma général de multiplication d'un état.

On suppose que l'état q doit être multiplié R_q fois. On note $\{e_1, ..., e_n\}$ l'ensemble des états entrants excepté q et $\{s_1, ..., s_m\}$ l'ensemble des états sortants excepté q. Les états d'éntrée et de sortie peuvent faire partie de ces deux ensembles.

Les connexions concernées par la multiplication de l'état q sont $(a_{e_i,q})_{1\leqslant i\leqslant n}, a_{qq}, (a_{q,s_i})_{1\leqslant i\leqslant m}$.

Les connexions du modèle modifié sont notées :

$$\left(a'_{e_i,q_k}\right)_{\substack{1\leqslant i\leqslant n\\1\leqslant k\leqslant R_q}}, \left(a'_{q_k,q_l}\right)_{\substack{1\leqslant k\leqslant R_q\\1\leqslant l\leqslant R_q}}, \left(a'_{q,s_i}\right)_{\substack{1\leqslant i\leqslant m\\1\leqslant k\leqslant R_q}}$$

$$\text{où } \left\{q_1, ..., q_{R_q}\right\} \text{ sont les } R_q \text{ multiplications de l'état } q.$$

Les probabilités d'émissions repectent les mêmes notations, celles concernées par cette modification sont $(b_q(o))_{1\leqslant o\leqslant O}$. Les probabilités d'émissions du modèle modifié sont notées $\left(b'_{q_k}(o)\right)_{\substack{1\leqslant o\leqslant O\\1\leqslant k\leqslant R_q}}$.

Le modèle obtenu est équivalent au premier si les probabilités d'émissions et de transitions de la copie sont identiques à celles de l'état copié et si les probabilités de transiter vers l'état d'origine sont partagées avec sa copie. Ce principe de multiplication sera repris pour chacune des trois méthodes d'extension de l'architecture qui suivent.

F.3.2 Introduction de cycles

F.3.2.1 Principe

La longueur des séquences qu'un modèle de Markov caché doit apprendre est inconnue, on sait seulement que cette longueur est finie. Lorsqu'un modèle ne peut modéliser que des séquences limitées en longueur, la probabilité d'émission d'une séquence plus longue est nulle. Dans ces cas, les formules d'apprentissage de Baum-Welch ne peuvent prendre en compte l'information contenue dans ces séquences. Cette méthode de croissance consiste dans un premier temps à ajouter des états cycliques au modèle puis de réestimer ces

paramètres. De cette façon, toute séquence aura une probabilité non nulle quelle que soit sa longueur. Dans un second temps, les états cycliques seront répliqués de manière à supprimer ces cycles tout en conservant leur aptitude à modéliser de longues séquences.

Pour un état donné i, on note $p = \mathbb{P}(i \in cycle)$ défini en (F.7), on suppose que $p < 1$. Dans le cas contraire, l'état est récurrent et ce cas ne peut se produire pour les modèles utilisés qui ne reconnaissent que des séquences finies d'observations. On note également $p' = 1-p$ la probabilité de sortir de l'état i (figure F.9).

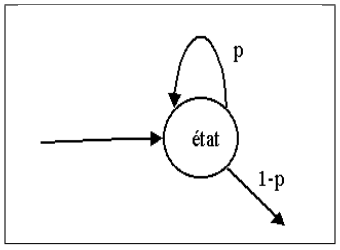


Fig. F.9: Schématisation d'un état appartenant à un cycle.

Si $p = 0,9$, ceci signifie que la séquence d'états que le modèle a apprise contient 9 transitions de cet état i vers lui-même avant de transiter vers un autre état ou la sortie. Il faudrait donc multiplier l'état i par $10 = \frac{1}{1-0,9}$ fois. Les copies de l'état i doivent suivre les deux règles suivantes :

1. La probabilité d'appartenir à un cycle dans le modèle contenant les copies est nulle.
2. La probabilité de sortie des copies est identique à celle de i.

Par conséquent, si N est le nombre de copies de l'état à créer, on trouve que : $(1-p) + N(1-p) = 1$, d'où le nombre de copies de l'état i à créer est :

$$N = \frac{p}{1-p} = \frac{\text{ce qui devrait sortir}}{\text{ce qui sort}}$$

Par conséquent, l'état i sera multiplié un nombre de fois égal à r_i avec :

$$r_i = \begin{cases} 0 \text{ si } \sum_{t=1}^{+\infty} c_t^i = 0 \\ \left[\frac{1}{(1-\mathbb{P}(i \in cycle))}\right] = \left[\frac{1}{\left(1-\sum_{t=1}^{+\infty} c_t^i\right)}\right] \text{ si } \sum_{t=1}^{+\infty} c_t^i > 0 \\ \text{n'est pas défini si } \sum_{t=1}^{+\infty} c_t^i = 1, \text{ dans ce cas, l'état } i \text{ est } \textbf{récurrent} \end{cases} \tag{F.23}$$

F.3.2.2 Un exemple illustration de la probabilité d'appartenir à un cycle

Le modèle figure F.10 est un modèle qui contient deux cycles, le premier inclut les états 0 et 1, sa durée est de deux, le second inclut les états 0,1,2, sa durée est de trois. On veut calculer la probabilité pour chacun des états d'appartenir à un cycle :

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14
c_t^0	0	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0	0	0	0	0	0	0	0	0

$$\mathbb{P}(0 \in cycle) = \tfrac{1}{2} + \tfrac{1}{4} = \tfrac{3}{4}$$

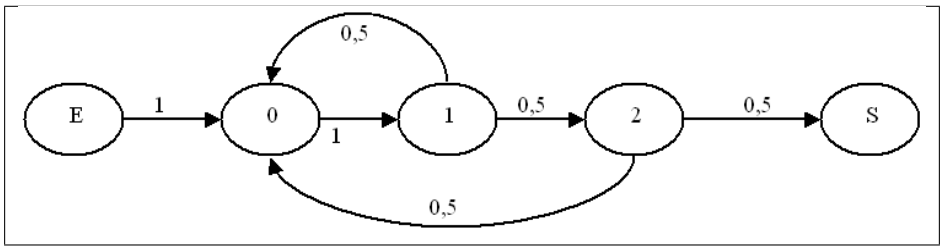


Fig. F.10: Modèle ayant appris la séquence 0101201012.

L'état 0 sera donc multiplié 4 fois.

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14
c_t^1	0	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0	0	0	0	0	0	0	0	0

$$\mathbb{P}\left(1 \in cycle\right) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$$

L'état 1 sera donc multiplié 4 fois.

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14
c_t^2	0	0	$\frac{1}{4}$	0	$\frac{1}{8}$	0	$\frac{1}{16}$	0	$\frac{1}{32}$	0	$\frac{1}{64}$	0	$\frac{1}{128}$	0

$$\mathbb{P}\left(3 \in cycle\right) = \frac{1}{4}\sum_{i=1}^{+\infty}\frac{1}{2^i} = \frac{1}{2}$$

L'état 2 sera donc multiplié 2 fois.

Par conséquent, cette méthode estime à 10 le nombre d'états nécessaire afin d'éviter les états semi-récurrents ce qui est dans ce cas l'exacte réponse puisque la séquence que le modèle a appris contient 10 observations. Voyons maintenant comment obtenir ces états.

F.3.2.3 Mise en place

Les notations utilisées sont celles du paragraphe F.3.1. Le nouveau modèle M' ne doit plus contenir de cycles, les transitions et émissions indéterminées doivent pour cela vérifier les conditions du tableau F.3. Ces formules ne suppriment pas les cycles mais dans le cadre de la reconnaissance de l'écriture, il est peu probable qu'ils subsistent. La figure F.11 illustre un exemple pour lequel $R_q = 3$.

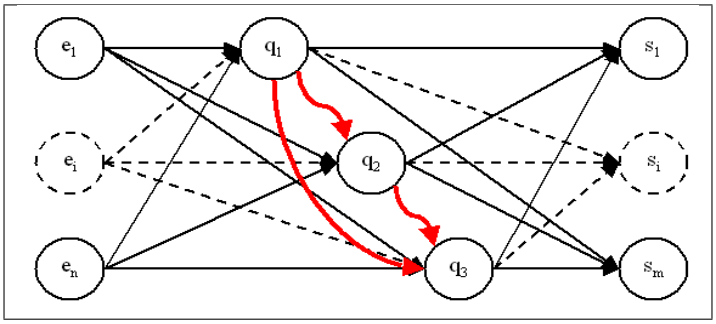


Fig. F.11: Suppression d'un cycle de longueur trois.

$$
\begin{array}{l}
\forall k \in \{1, \cdots, R_q\}, \\
\forall l \in \{1, \cdots, R_q\}, \\
\forall o \in \{1, \cdots, O\}, \\
\forall i \in \{1, \cdots, n\}, \\
\forall j \in \{1, \cdots, m\},
\end{array}
\begin{array}{rcl}
b'^{q_k}(o) & = & b_q(o) \\
a'_{q_k,q_l} & = & 0 \text{ si } k \geqslant l \\
a'_{q_k,q_l} & = & \dfrac{\mathbb{P}(q \in cycle)}{R_q - k} \text{ si } k < l \\
a'_{e_i,q} & = & 0 \text{ car l'état } q \text{ est détruit} \\
a'_{e_i,q_k} & = & \dfrac{a_{e_i,q}}{R_q} \\
a'_{q_k,s_m} & = & \dfrac{1 - \mathbb{P}(q \in cycle)}{R_q - k}
\end{array}
$$

Tab. F.3: Nouvelles valeurs des coefficients lors de la suppression des cycles.

F.3.3 Association d'un état à une classe d'observations

F.3.3.1 Principe

Nous avons vu que la probilité d'émission d'une observation sachant un état s'exprime comme une somme de probabilités incluant celles d'appartenance d'un graphème à chacune des classes de sortie du réseau de neurones ou d'un autre classifieur [2]. L'objectif est de réduire cette somme complexe afin de limiter les temps de calcul. Plutôt que d'avoir un état susceptible d'émettre des observations appartenant à plusieurs classes différentes, il est préférable d'avoir plusieurs états chacun capable d'émettre des observations n'appartenant qu'à une seule classe (figure F.12).

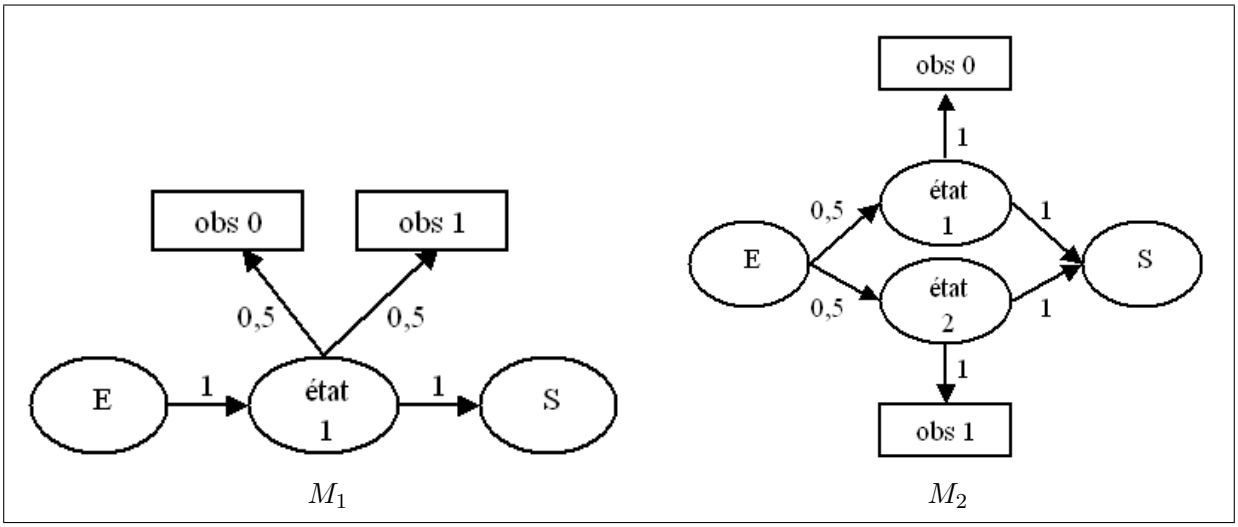


Fig. F.12: Deux modèles équivalents d'architectures différentes. Dans les deux cas, les deux séquences d'observations constituées chacunes d'un seul 0 ou d'un seul 1 autant probables avec l'un ou l'autre des modèles. Le second modèle, en associant états et observations sans ambiguïtés, est plus lisible. Il permet de mieux "décoder" la séquence par un alignement Viterbi.

2. Annexes : voir paragraphe E.5, page 281

Considérons les deux chaînes de Markov cachées M_1 et M_2 représentées par la figure F.12. On note O_0 la séquence (0) (une seule observation) et O_1 la séquence (1) (une seule observation également), on obtient que :

$$\mathbb{P}(O_0 \mid M_1) = \mathbb{P}(O_0 \mid M_2) = \mathbb{P}(O_1 \mid M_1) = \mathbb{P}(O_1 \mid M_2) = 0,5$$

Cependant, lors d'un alignement Viterbi, le modèle M_1 retournera le même chemin pour les deux séquences O_0 et O_1 alors que le modèle M_2 retournera deux chemins différents, donc plus d'information. C'est une raison pour laquelle il est préférable que les probabilités d'émission d'un état soient centrées autour d'une seule classe d'observations. Cette association entre état et classe nécessite de dupliquer des états pour obtenir des modèles équivalents. On cherche donc à multiplier les états i pour lesquels $o \neq o'$, $b_i(o)$ et $b_i(o')$ sont du même ordre.

Soient K séquences d'observations notées $O = \left(O^1, \ldots, O^K\right)$ avec $O^k = \left(O_1^k, \ldots, O_{T_k}^k\right)$. On définit la probabilité $P_{in} = \mathbb{P}(C_n | q_t = i, O)$ la probabilité de la classe n sachant l'état i :

$$\begin{aligned}
P_{in} &= \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbb{P}\left(C_n | q_t = i, O^k\right) \mathbb{P}\left(q_t = i, O^k\right) \\
&= \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbb{1}_{\left\{O_t^k = C_n\right\}} \alpha_t^k(i)\, \beta_t^k(i) \quad \text{pour des observations discrètes} \qquad \text{(F.24)} \\
&= \sum_{k=1}^{K} \sum_{t=1}^{T_k} \frac{\mathbb{P}\left(O_t^k | C_n\right) \mathbb{P}(C_n | q_t = i)}{\sum\limits_{l} \mathbb{P}\left(O_t^k | C_l\right) \mathbb{P}(C_l | q_t = i)}\, \alpha_t^k(i)\, \beta_t^k(i) \quad \text{pour des observations continues} \qquad \text{(F.25)}
\end{aligned}$$

On définit pour $s \in [0,1]$, S_i^s est l'ensemble :

$$S_i^s = \min \left\{ n \left| \frac{P_{in}}{\max\limits_k P_{ik}} \geqslant s \right. \right\} \qquad \text{(F.26)}$$

F.3.3.2 Mise en place

Si l'ensemble $S_i^s = \left(S_i^s(1), \ldots, S_i^s\left(N_i^s\right)\right)$ contient plus d'un élément, alors il sera multiplié selon les règles du tableau F.4. La figure F.13 illustre un exemple pour $N_i^s = 3$.

F.3.4 Transitions d'ordre supérieur à un

F.3.4.1 Principe

Le théorème E.6.9 a montré qu'un modèle d'ordre n est équivalent à un modèle d'ordre un contenant un plus grand nombre d'états. Il est alors possible de définir pour chaque ordre n, un nombre d'états D_n correspondant au nombre minimal d'états d'une chaîne de Markov cachée d'ordre n ayant appris un ensemble de séquences d'observations. Etant donné que les modèles utilisés lors de la reconnaissance de l'écriture sont tous d'ordre un, que se passe-t-il si celui trouvé grâce à la méthode présentée au paragraphe F possède un nombre d'états strictement inférieur à D_1 ? Ce paragraphe propose une méthode pour déceler ce cas.

$$
\begin{array}{l}
\forall k \in \{1, \cdots, N_i^s\}, \\
\forall l \in \{1, \cdots, N_i^s\}, \\
\forall o \in \{1, \cdots, O\}, \\
\forall i \in \{1, \cdots, n\}, \\
\forall j \in \{1, \cdots, m\},
\end{array}
\begin{array}{lcl}
b'^{q_k}(o) & = & \mathbb{1}\!\!1_{\{o=S_i^s(k)\}} \\
a'_{q_k,q_l} & = & 0 \text{ si } k \neq l \\
a'_{q_k,q_l} & = & a_{q,q} \text{ si } k = l \\
a'_{e_i,q} & = & 0 \text{ car l'état } q \text{ est détruit} \\
a'_{e_i,q_k} & = & a_{e_i,q} \\
a'_{q_k,s_m} & = & a_{q,s_m}
\end{array}
$$

Tab. F.4: Nouvelles valeurs des coefficients lors de l'association d'un état à une classe

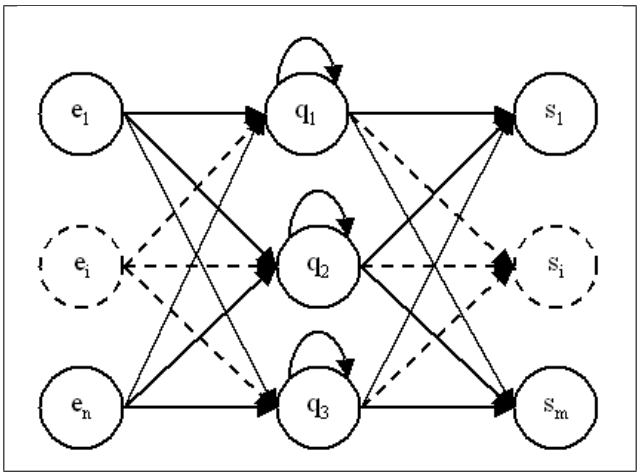


Fig. F.13: Multiplication des états pour les associer à une classe

Elle est fondée sur l'idée que le modèle appris M ayant N états n'en contient pas assez et que le modèle optimal ayant ce même nombre d'états N est d'ordre supérieur à un. On note $O = \left(O^1, \ldots, O^K\right)$ l'ensemble des séquences que le modèle a apprises avec $\forall k,\ O_k = \left(O_1^k, \ldots, O_{T_k}^k\right)$, on note S l'ensemble des séquences d'états et $s = (q_1, \ldots, q_T)$ une séquence d'états de longueur T, $\overline{q_t} = (q_1, \ldots, q_t)$. Il s'agit donc de vérifier que :

$$\forall s \in S,\ \mathbb{P}\left(q_t \mid q_{t-1}, O, M\right) = \mathbb{P}\left(q_t \mid \overline{q_{t-1}}, O, M\right)$$

Afin de simplifier cette tâche, on pose $d \geqslant 2$ pour vérifier que :

$$
\begin{aligned}
\mathbb{P}\left(q_t \mid q_{t-1}, O, M\right) &= \mathbb{P}\left(q_t \mid q_{t-1}, \ldots, q_{t-d}, O, M\right) \\
&= \mathbb{P}\left(q_t \mid q_{t-d}^{t-1}, O, M\right)
\end{aligned}
$$

Par conséquent, si i et j sont fixés, la probabilité $\mathbb{P}\left(q_t = j \mid q_{t-1} = i, q_{t-2}, \ldots, q_{t-d}, O, M\right)$ doit être constante et c'est cette hypothèse qui sera testée. Au préalable, comme M est supposé être d'ordre un, ses probabilités de transition sont notées :

$$(a_{ij})_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}} = (\mathbb{P}(q_t = j \mid q_{t-1} = i, M))_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}} \tag{F.27}$$

Et on définit l'ensemble S_{ij}^d de séquences d'états de probabilités non nulles menant aux états i et j :

$$\begin{gathered} S_{ij}^d = \left\{ (q_1, \ldots, q_{d+1}) \;\middle|\; \begin{array}{l} q_d = i, q_{d+1} = j \text{ et} \\ \mathbb{P}(q_1, \ldots, q_{d+1}|M) > 0 \end{array} \right\} \\ \text{où } \mathbb{P}(q_1, \ldots, q_{d+1}|M) = \prod_{t=1}^{d} a_{q_t q_{t+1}} \end{gathered} \tag{F.28}$$

Les élements S_{ij}^d sont notés $S_{ij}^d = \left\{ s_1^{ij}, \ldots, s_{m_{ij}^d}^{ij} \right\}$ et s_n^i est la séquence s_n^{ij} privée de l'état indice $d+1$. On note le vecteur $V_{ij}^d = \left(V_{ij}^d(1), \ldots, V_{ij}^d\left(m_{ij}^d\right) \right)$ défini par :

$$\forall n, \quad V_{ij}^d(n) = \begin{cases} 0 \text{ si } \sum_{k=1}^{K} \sum_{t=d+1}^{T_k} \mathbb{P}\left(q_{t-d}^t = s_n^i \mid O^k, M\right) = 0 \\ \dfrac{\sum_{k=1}^{K} \sum_{t=d+1}^{T_k - 1} \mathbb{P}\left(q_{t-d}^{t+1} = s_n^{ij} \mid O^k, M\right)}{\sum_{k=1}^{K} \sum_{t=d+1}^{T_k} \mathbb{P}\left(q_{t-d}^t = s_n^i \mid O^k, M\right)} \text{ sinon} \end{cases} \tag{F.29}$$

Si le modèle M est supposé être d'ordre un alors les éléments du vecteur V_{ij}^d sont tous identiques, il vérifie donc :

$$\forall n \in \left\{1, \cdots, m_{ij}^d\right\}, \; \frac{V_{ij}^d(n)}{\sum_{n=1}^{m_{ij}^d} V_{ij}^d(n)} = \frac{1}{m_{ij}^d} \tag{F.30}$$

Le vecteur V_{ij}^d est donc l'estimation des paramètres d'une loi multinomiale de dimension m_{ij}^d dont la distribution attendue est uniforme. On construit ainsi la statistique de test suivante :

$$X_{ij}^d = K_{ij} \sum_{n=1}^{m_{ij}^d} m_{ij}^d \left[\frac{V_{ij}^d(n)}{\sum_{n=1}^{m_{ij}^d} V_{ij}^d(n)} - \frac{1}{m_{ij}^d} \right]^2 \tag{F.31}$$

$$\text{avec } K_{ij} = \sum_{k=1}^{K} \sum_{t=2}^{T_k} \mathbb{P}\left(q_t = j, q_{t-1} = i | O^k, M\right)$$

Pour chaque transition, on effectue le test suivant :

Hypothèse 0 : La transition de l'état i vers l'état j à l'instant t ne dépend que de l'état i ou des états aux instants $t-d-1$ et antérieurs.

Hypothèse 1 : La transition de l'état i vers l'état j à l'instant t dépend des états aux instants compris entre $t-1$ et $t-d$.

Si l'hypothèse nulle est vraie, alors X_{ij}^d suit une une loi χ_2 à $m_{ij}^d - 1$ degrés de liberté[3]. Si t_α vérifie $\mathbb{P}\left(X_{ij}^d \leqslant t_\alpha\right) = \alpha$, ceci nous permet de définir l'ordre Or_{ij}^α de la transition $i \longrightarrow j$ comme étant :

$$Or_{ij}^\alpha = \begin{cases} 1 \text{ si } \forall d \geqslant 2,\ X_{ij}^d \leqslant t_\alpha \\ \min\left\{d \geqslant 2 \mid X_{ij}^d \leqslant t_\alpha\right\} \text{ sinon} \end{cases} \tag{F.32}$$

Si le modèle M est effectivement d'ordre un, pour chaque transition $i \longrightarrow j$, et pour chaque d, l'hypothèse nulle de ce test doit être validée. Cette méthode est applicable dans le domaine de la reconnaissance de l'écriture car :

1. Les modèles ne contiennent pas de cycle (la probabilité de transiter d'un état vers un autre déjà visité est nulle).
2. Les séquences sont courtes (une dizaine de d'observations), donc la valeur maximale de d est petite.
3. Le nombre d'états des modèles est de l'ordre de quelques dizaines.

Pour ces trois raisons, ce test est applicable aux modèles utilisés pour la reconnaissance de l'écriture avec un temps de calcul raisonnable.

F.3.4.2 Exemple

Les classes d'observations possibles sont cette fois-ci l'ensemble $\{0,1,2,3\}$ et le modèle M_3 doit apprendre les séquences "020", "121", "021", "120", "323", "321", chacune d'elles dix fois. Le modèle M_3 obtenu en appliquant la méthode de sélection des transitions est représenté figure F.14. Les statistiques de tests obtenus pour les transitions $4 \longrightarrow 5$, $4 \longrightarrow 6$, $4 \longrightarrow 7$ sont précisés dans le tableau F.5.

transition	m_{ij}^3	K_{ij}	X_{ij}^3	$t \vert \mathbb{P}\left(X_{ij}^3 \leqslant t\right) = 0,95$
$4 \longrightarrow 5$	3	20	10	8,15
$4 \longrightarrow 6$	3	30	0	8,15
$4 \longrightarrow 7$	3	10	16,7	8,15

Tab. F.5: Statistiques de test pour le modèle M_3.

Le remède le plus simple apporté au modèle lorsqu'est détectée une transition d'ordre supérieur à un est de créer des états. Si l'estimation de l'ordre de la transition $i \longrightarrow j$ est $d \geqslant 2$, alors les états i et tous ceux qui peuvent l'atteindre en $d-2$ transitions au plus sont dupliqués selon les règles (F.33) à (F.39).

Le modèle est ensuite réappris et converge vers le modèle M_4 (figure F.15) pour lequel les statistiques de tests sont toutes nulles. Dans ce cas simple, le modèle M_4 est optimal pour les séquences d'observations qu'il doit apprendre comme le confirme le tableau F.6.

3. Soit une variable aléatoire $X \in \{1,\ldots,N\}$. L'estimation de sa distribution réalisée à partir d'un échantillon de T exemples indépendants et de même loi. On suppose que la distribution empirique de X est $(p_1,\ldots,p_N)$. On veut tester si X suit la loi multinomiale de distribution $(q_1,\ldots,q_N)$. On calcule $d^2 = T \sum_{i=1}^{N} \frac{(p_i-q_i)^2}{q_i}$. Sous l'hypothèse que X suit la loi multinomiale de distribution $(q_1,\ldots,q_N)$, la loi limite de d^2 lorsque $T \longrightarrow +\infty$ est une loi χ^2_{N-1} à $N-1$ degrés de liberté (voir [Saporta1990]).

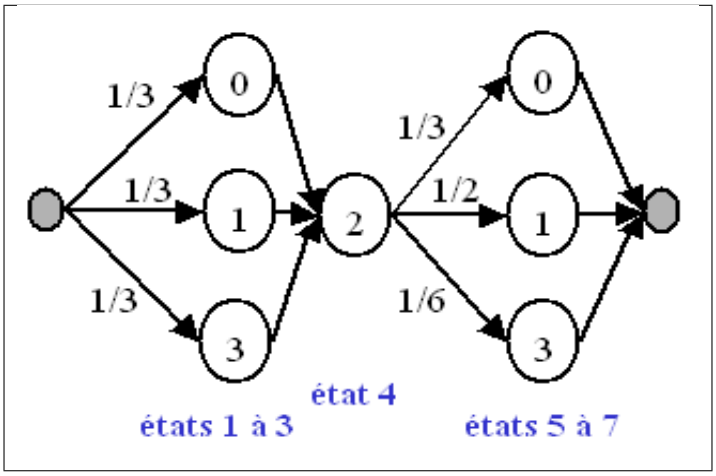


Fig. F.14: Modèle M_3 obtenu après suppression des transitions faibles.

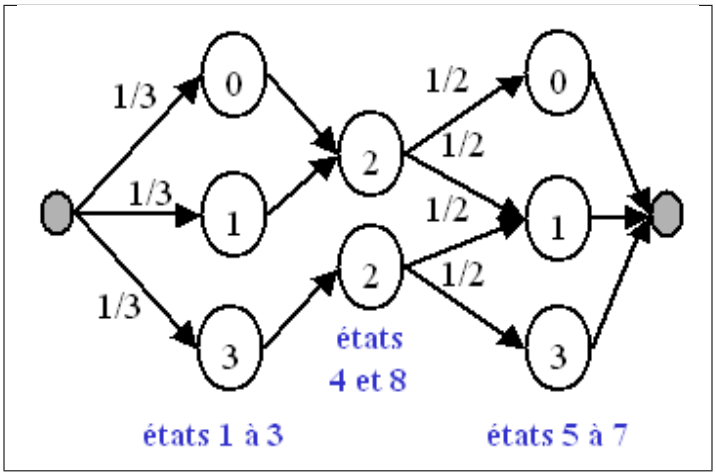


Fig. F.15: Modèle M_4 finalement obtenu.

F.3.4.3 Justification

Soit M une chaîne de Markov cachée à N états ne contenant pas de cycle. On note $(A_{ij})_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}}$ sa matrice de transition. Soit $u \in \{1, \cdots, N\}$. On note $A^u = \{i | a_{iu} > 0\}$ et $B^u = \{j | a_{uj} > 0\}$. A^u et B^u sont l'ensemble des prédécesseurs et des successeurs de l'état k. On suppose également que ni l'état de sortie, ni l'état d'entrée n'appartiennent à ces deux ensembles (voir figure F.16). La chaîne de Markov cachée a appris les séquences $O = \left(O^1, \ldots, O^K\right)$ avec $\forall k \in \{1, \cdots, K\}$, $O^k = \left(O_1^k, \ldots, O_{T_k}^k\right)$.

On construit la chaîne de Markov M' contenant $N+1$ d'états, $\left(a'_{ij}\right)_{\substack{1\leqslant i\leqslant N\\1\leqslant j\leqslant N}}$ sont ses probabilités de transition. Les probabilités d'émissions de l'état $N+1$ sont les mêmes que celles de l'état u, de plus pour $x \in B^u$, on a (voir figure F.17) :

séquences d'observations	$\mathbb{P}(\mathbf{O}\|\mathbf{M_3})$	$\mathbb{P}(\mathbf{O}\|\mathbf{M_4})$
020	1/9	1/6
021	1/6	1/6
120	1/9	1/6
121	1/9	1/6
321	1/9	1/6
323	1/18	1/6
somme	14/18	1
séquences non apprises	$\mathbb{P}(\mathbf{O}\|\mathbf{M_3})$	$\mathbb{P}(\mathbf{O}\|\mathbf{M_4})$
023	1/18	0
123	1/18	0
320	1/9	0
somme	4/18	0

Tab. F.6: Probabilités des séquences M_3 et M_4.

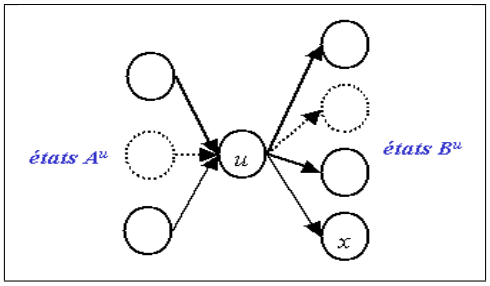


Fig. F.16: Modèle M initial

$$a'_{ij} = a_{ij} \text{ si } i \neq u \text{ et } j \neq u \tag{F.33}$$
$$a'_{iu} = a_{iu}(1 - a_{ux}) \tag{F.34}$$
$$a'_{i,N+1} = a_{iu}a_{ux} \tag{F.35}$$
$$a'_{ui} = \frac{a_{ui}}{1 - a_{ux}} \tag{F.36}$$
$$a'_{ux} = a'_{u,N+1} = a'_{N+1,u} = 0 \tag{F.37}$$
$$a'_{N+1,i} = 0 \text{ si } i \neq x \tag{F.38}$$
$$a'_{N+1,x} = 1 \tag{F.39}$$

Il est évident que : $\forall k,\ \mathbb{P}\left(O^k|M\right) = \mathbb{P}\left(O^k|M'\right)$. Comme le modèle M a appris les séquences d'observations $\left(O^1, \ldots, O^K\right)$, on note $P_k^M = \mathbb{P}\left(O^k|M\right)$, alors ([Levinson1983]) :

– la vraisemblance $L(O, M)$ est maxiumum

– $a_{iu} = \overline{a_{iu}} = \dfrac{\sum_{k=1}^{K}\sum_{t=2}^{T_k}\mathbb{P}\left(q_t = u, q_{t-1} = i|O^k, M\right)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbb{P}\left(q_t = i|O^k, M\right)}$

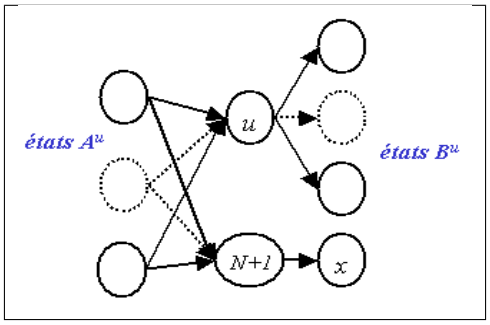


Fig. F.17: Modèle M' dérivé de M

On suppose maintenant que $X^3_{uv} > 0$, la réestimation du coefficient $a'_{i,N+1}$ par les formules de Baum-Welch donne :

$$\overline{a'_{i,N+1}} = \frac{\sum_{k=1}^{K}\sum_{t=2}^{T_k}\mathbb{P}\left(\begin{smallmatrix}q_t=N+1\\q_{t-1}=i\end{smallmatrix}\middle| O^k, M'\right)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbb{P}\left(q_t = i|O^k, M'\right)}$$

$$\overline{a'_{i,N+1}} = \frac{\sum_{k=1}^{K}\sum_{t=2}^{T_k-1}\mathbb{P}\left(\begin{smallmatrix}q_{t+1}=x\\q_t=u\\q_{t-1}=i\end{smallmatrix}\middle| O^k, M\right)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbb{P}\left(q_t = i|O^k, M\right)}$$

$$\overline{a'_{i,N+1}} = \overline{a_{iu}}\,\frac{\sum_{k=1}^{K}\sum_{t=2}^{T_k-1}\mathbb{P}\left(\begin{smallmatrix}q_{t+1}=x\\q_t=u\\q_{t-1}=i\end{smallmatrix}\middle| O^k, M\right)}{\sum_{k=1}^{K}\sum_{t=2}^{T_k}\mathbb{P}\left(\begin{smallmatrix}q_t=u\\q_{t-1}=i\end{smallmatrix}\middle| O^k, M\right)}$$

Si $X^3_{ux} > 0$, alors $\exists i \in A^u$ tel que : $\frac{\overline{a'_{i,N+1}}}{a_{iu}} \neq a_{ux}$. Par conséquent, après une itération de l'algorithme Baum-Welch, $L\left(O^1, \ldots, O^K, M'\right) > L\left(O^1, \ldots, O^K, M\right)$. En revanche, si $X^3_{ux} = 0$, la vraisemblance de ce modèle ne sera pas augmentée, cette remarque laisse supposer que la réciproque est vraie. Cette démonstration peut être étendue au cas $d > 2$ en supposant que : $\forall d' < d,\ X^{d'}_{ux} = 0$.

F.3.4.4 Mise en pratique

Celles-ci sont énoncées par les régles (F.33) à (F.39) dans le cas où $d = 2$. Dans le cas où $d > 2$, plusieurs états doivent être dupliqués, soit l'ensemble D_u suivant :

$$D_u = \left\{q \text{ tels que } \exists d' \in \{1, \cdots, d-2\},\ \mathbb{P}\left(q_t|q_{t-d'}\right) > 0\right\}$$

Chaque état de l'ensemble D_u va être dupliqué, on note q un élément de D_u et $\overline{q}$ sa copie, E_u est le complémentaire de D_u. Les règles de mise à jour des coefficients sont données par le tableau F.7.

Cette méthode est néanmoins assez coûteuse en calcul même si, dans le cas de la reconnaissance de l'écriture manuscrite, les valeurs de d n'excède pas 3 ou 4.

$$
\begin{array}{rl}
\forall q \in D_u, & \\
\forall q_2 \in D_u, & \\
\forall p \in E_u, & \\
\forall p_2 \in E_u, & \\
a'_{q\,q_2} &= a_{q\,q_2} \text{ si } q \neq u \text{ et } q_2 \neq u \\
a'_{q\,q_2} &= \dfrac{a_{u\,q_2}}{1 - a_{u\,i}} \mathbb{1}_{\{q_2 = i\}} \text{ si } q = u \\
a'_{\overline{q}\,\overline{q_2}} &= a_{q\,q_2} \text{ si } \overline{q} \neq u \text{ et } \overline{q_2} \neq u \\
a'_{\overline{q}\,\overline{q_2}} &= \mathbb{1}_{\{q_2 = i\}} \text{ si } q = u \\
a'_{\overline{q}\,q_2} &= 0 \\
a'_{q\,\overline{q_2}} &= 0 \\
a'_{p\,p_2} &= a_{p\,p_2} \\
a'_{p\,q_2} &= (1 - a_{u\,i})\, a_{p\,q_2} \\
a'_{p\,\overline{q_2}} &= a_{u\,i} a_{p\,q_2} \\
a'_{q\,p_2} &= a_{q\,p_2} \\
a'_{\overline{q}\,p_2} &= a_{q\,p_2}
\end{array}
$$

Tab. F.7: Règles associées à l'ajout d'un état dans un modèle afin de diminuer l'ordre des transitions.

F.4 Sélection automatique de l'architecture

L'objectif de la reconnaissance de l'écriture est de trouver des modèles de Markov cachés modélisant au mieux un ensemble de séquences finies et il n'est pas toujours possible de connaître le nombre d'états cachés nécessaires à cette modélisation. Jusqu'à présent, ce nombre d'états a toujours été fixé et toutes les transitions d'un état à un autre sont possibles. Dans le cas de l'exemple du joueur trichant à l'aide de pièce de monnaie truquée ou non truquée (paragraphe E.2.1, page 247), cela suppose que le nombre de pièces différentes qu'il posséde est connu, ce qui peut ne pas être le cas.

Les six méthodes d'évolution de l'architecture présentées dans les paragraphes précédents sont agencées de manière à former un algorithme de sélection d'architecture. Il alterne période de croissance et décroissance jusqu'à ce que les performances se stabilisent.

Après une modification de structure, il est nécessaire de réestimer les paramètres des modèles. Cette étape est assurée par l'algorithme E.4.2. Les paramètres nuls insérés lors de la modification de la structure ne

doivent pas non plus le rester, il suffit d'appliquer l'astuce décrite au paragraphe E.4.8.

Algorithme F.4.1 : sélection d'architecture
Etape A : initialisation
Le point de départ peut être soit un modèle réduit à un état cyclique capable d'émettre toutes les classes d'observations ou alors un modèle exhaustif comprenant suffisamment de connexions pour modéliser la majorité des séquences d'observations qu'il doit apprendre. Ces deux initialisations sont schématisées par la figure F.7. Dans le cas de la reconnaissance de l'écriture, la seconde solution semble meilleure et c'est d'ailleurs celle préconisée par [Augustin2001] (voir paragraphe F.2.1). Après l'estimation ce modèle exhautif, l'initialisation se poursuit par une suppression des connexions inutiles.

Etape B : introduction de cycle
1. utilisation de la méthode introduisant des états cycliques
2. réestimation
3. à nouveau, suppression des connexions inutiles
4. réestimation

Etape C : un état, une classe d'observation
1. utilisation de la méthode associant à chaque état une seule classe d'observation
2. réestimation
3. à nouveau, suppression des connexions inutiles
4. réestimation

Etape D : étape facultative : réduction de l'ordre des transition
1. utilisation de la méthode réduisant l'ordre des transitions
2. réestimation
3. à nouveau, suppression des connexions inutiles
4. réestimation

Etape E : terminaison
Tant que les performances s'améliorent, on retourne à l'étape B, sinon, l'algorithme s'arrête. Les résultats peuvent être affinés en utilisant le regroupement d'états similaires suivi d'une réestimation.

Remarque F.4.2: sélection dans le cadre de la reconnaissance
L'étape D est facultative car trop coûteuse comparée au gain en performances en ce qui concerne la reconnaissance de l'écriture. Il en est de même pour la méthode de regroupement des états similaires dont le coût est très élevé pour le nombre de connexions supprimées comparé à la méthode simple de suppression des connexions.

Annexe G

Modèles de Markov cachés et graphes d'observations

Le chapitre 4.5 montre comment sont utilisés les modèles de Markov cachés dans la reconnaissance de l'écriture. Le paragraphe 4.7 présente une méthode dont l'obectif est de contourner le problème des mauvaises segmentations graphèmes. Elle s'appuie sur la modélisation des erreurs les plus fréquentes. Une autre direction possible est la représentation de la segmentation graphèmes sous forme de graphe. La figure G.1 illustre un cas où, plutôt que de prendre une décision afin d'obtenir une séquence de graphèmes, la segmentation aboutit à un graphe d'observations.

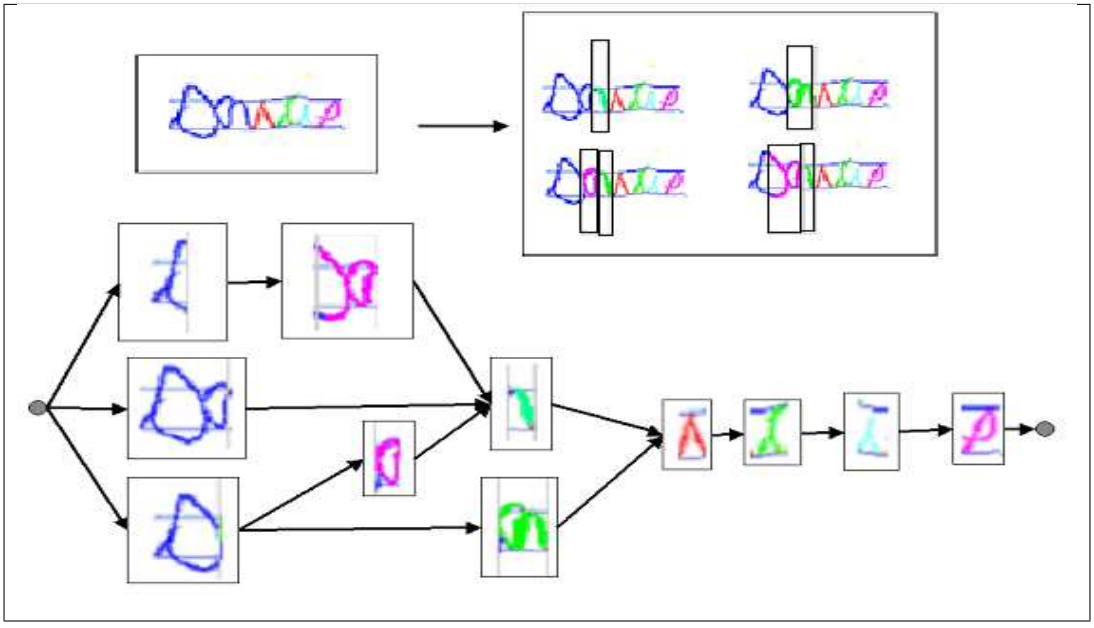


Fig. G.1: Exemple de graphe d'observations : aucune décision n'est prise concernant les trois lettres "sou" et leur segmentation est représentée sous forme de graphe, espérant que l'une des options proposées correspond à la bonne solution.

L'objectif de ce chapitre est de proposer l'extension des résultats obtenus dans l'annexe E et le chapitre 4.5 pour des séquences d'observations à des graphes d'observations. Les résultats s'inspirent de la thèse de [Saon1997a].

Ce chapitre se conclut sur la présentation d'un formalisme fondé sur les graphes (voir [Bengio1992]) et qui permet d'imbriquer plusieurs couches de modèles, de concevoir l'utilisation et l'estimation de ces modèles non plus de manière globale mais comme un enchaînement de graphes, à l'instar des couches de neurones à l'intérieur d'un réseau. Le calcul d'une probabilité propage les propabilités du premier graphe (proche de l'image) au dernier (proche du résultat final). L'estimation rétropropage ces probabilités en sens opposé.

G.1 Graphe d'observations

Soit $O = \{O_1, \ldots, O_G\}$ un ensemble d'observations, il s'agit de définir un graphe sur cet ensemble qui puisse être modélisable par une chaîne de Markov cachée. Etant donné que ces modèles imposent une dépendance temporelle d'une observation envers son passé, les graphes construits sur l'ensemble O ne peuvent pas être cycliques. Les graphes d'observations utilisés par la suite obéissent à la définition G.1.1. En outre, chaque arc de ce graphe sera pondéré par une probabilité déterminée par le prétraitement d'image.

Définition G.1.1 : graphe d'observations
Soit $O = \{O_1, \ldots, O_G\}$ un ensemble d'observations, O^1 est la première observation, O^f est la dernière observation, $O^{next}(O_i)$ désigne l'observation suivant O_i. On définit les vecteurs $\Pi^O = \left(\pi_i^O\right)_{1 \leqslant i \leqslant G}$, $\Theta^O = \left(\theta_i^O\right)_{1 \leqslant i \leqslant G}$ et la matrice $A^O = \left(a_{ij}^O\right)_{\substack{1 \leqslant i \leqslant G \\ 1 \leqslant j \leqslant G}}$ de telle sorte que :

$$\begin{array}{rrcl} \forall i, & \pi_i^O & = & \mathbb{P}\left(O^1 = O_i\right) \\ \forall i, & \theta_i^O & = & \mathbb{P}\left(O^f = O_i \mid O_i\right) \\ \forall i, \forall j, & a_{ij}^O & = & \mathbb{P}\left(O^{next}(O_i) = O_j \mid O_i\right) \end{array} \tag{G.1}$$

Les coefficients Π, Θ, A vérifient les conditions suivantes :

$$\sum_{i=1}^{G} \pi_i^O = 1 \text{ et } \forall i,\ \theta_i^O + \sum_{j=1}^{G} a_{ij}^O = 1 \tag{G.2}$$

Le graphe ne doit contenir aucun cycle, ce qui est équivalent à dire qu'il existe une permutation des lignes et des colonnes de la matrice A^O de telle sorte qu'elle soit triangulaire supérieure avec des zéros sur la diagonale.

Cette définition est semblable à celle d'une chaîne de Markov (E.1.1) avec la contrainte supplémentaire de ne contenir aucun cycle [1].

G.2 Probabilité d'un graphe d'observations

G.2.1 Séquences admissibles

Soit M un modèle de Markov caché d'ordre un vérifiant la définition E.2.1 (page 249), on note Π^M, Θ^M, A^M, B^M ses paramètres. Soit $G(O)$ un graphe d'observations sur l'ensemble $O = \{O_1, \ldots, O_G\}$, G désigne également le nombre de nœuds de ce graphe qui a pour paramètres Π^G, Θ^G, A^G. L'objectif est de construire la probabilité $\mathbb{P}(G(O) \mid M)$. Soit $S^M = \left(s_1^M, \ldots, s_s^M\right)$ une séquence d'états du modèle M, alors :

1. Annexes : voir paragraphe F.1.2, page 295

$$\mathbb{P}\left(G\left(O\right) \mid M\right) = \sum_{S^M} \mathbb{P}\left(G\left(O\right), S^M \mid M\right)$$

La différence par rapport aux chaînes de Markov cachée est que les séquences d'états admissibles pour le graphe $G\left(O\right)$ ne sont plus de longueur fixe. De plus, cette séquence q doit être associée à une séquence S^G de nœuds de G de même longueur s. Pour simplifier, on note deux séquences S^M et S^G, si elles sont admissibles ($S^M \mathcal{R} S^G$), alors le calcul de la probabilité $\mathbb{P}\left(G\left(O\right), S^M, S^G \mid M\right)$ est possible. Par conséquent :

$$\mathbb{P}\left(G\left(O\right) \mid M\right) = \sum_{S^M \mathcal{R} S^G} \mathbb{P}\left(S^M, S^G \mid M, G\right) \tag{G.3}$$

Cette formulation définit bien la probabilité cherchée mais ne permet pas de la calculer. L'objectif est de définir un algorithme similaire à l'algorithme *forward* (E.2.3). Toutefois, avant d'aller plus loin dans la description de cet algorithme, il est préférable de revenir sur ce que signifie cette modélisation au sens "physique".

G.2.2 Aparté : sens physique de la modélisation

La figure G.1 est sans ambiguïté en ce sens que le graphe des observations qu'elle décrit propose quatre différentes manières d'écrire le groupe de lettres "sou". Parmi ces quatre-là, on peut raisonnablement penser qu'une seule est bonne, c'est-à-dire que parmi les différentes séquences d'observations incluses dans ce graphe, il en existe une qui correspond à la véritable segmentation en graphèmes.

Intéressons-nous maintenant au problème des accents. Ces derniers se baladent fréquemment au-dessus de la lettre accentuée mais empiètent parfois un peu ou complètement sur l'espace aérien de ses voisines comme le montre la figure G.2.

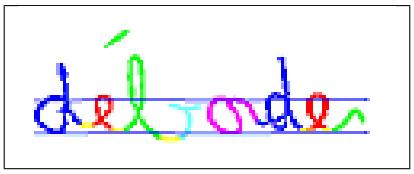


Fig. G.2: Accent empiétant sur l'espace aérien de la lettre "b", comment représenter le graphe de segmentation du couple "éb".

Il est possible de réduire le problème de segmentation au couple de lettres "éb". Le formalisme induit par la formule (G.3) considère qu'il existe une seule bonne séquence d'observations correspondant au couple "éb", celle-ci est incluse dans le graphe d'observations proposé par la figure G.3. L'accent est associé à une lettre pour former un graphème mais n'apparaît pas seul.

Une seconde option de représentation est celle de la figure G.4. Plus proche spatialement de l'image, certaines séquences d'observations ne contiennent cependant pas d'accent. Toutefois, est-ce que cette modélisation est en soit inadaptée à la modélisation du paragraphe précédent ? Un rapide retour à l'objectif de la reconnaissance nous permet de trouver des éléments de réponse à cette question. S'il importe de distinguer les lettres accentuées de celles qui ne le sont pas, voyons si les deux graphes d'observations (figures G.3, et G.4) permettent la distinction des deux groupes de lettres "eb" et "éb".

Le graphe de la figure G.3 inclut l'accent dans un des deux graphèmes de lettres, par conséquent, la forme

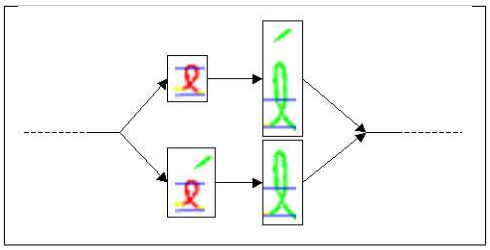


Fig. G.3: Accent empiétant sur l'espace aérien de la lettre "b" (voir figure G.2) : graphe d'observations relatif au couple "éb" incluant la véritable séquence d'observations. L'accent est associé à une lettre pour former un graphème mais n'apparaît pas seul.

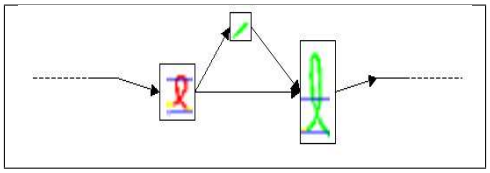


Fig. G.4: Accent empiétant sur l'espace aérien de la lettre "b" (voir figure G.2) : graphe d'observations relatif au couple "éb" n'incluant pas la véritable segmentation mais plus proche d'une représentation spatiale de la segmentation.

des graphèmes permet de distinguer une lettre accentuée. Cette tâche est celle d'un classifieur [2]. Le graphe de la figure G.4 impose l'accent dans un graphème supplémentaire. Cette différence génère des confusions pour les modèles "eb" et "éb" puisque tous deux ont maintenant une séquence en commun :

– le modèle "eb" modélise la séquence de graphèmes "eb"
– le modèle "éb" modélise les deux séquences de graphèmes "eb" et "e/b"

Pour éviter ce genre de confusion, il faudrait construire un graphe d'observations comme celui de la figure G.5 mais cette schématisation est trop stricte pour des accents qui aiment se promener parfois loin de leur lettre d'attache.

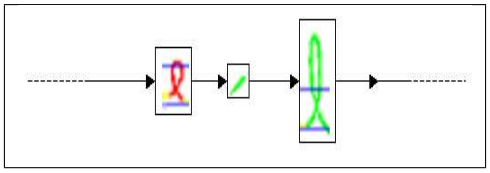


Fig. G.5: Accent empiétant sur l'espace aérien de la lettre "b" (voir figure G.2) : graphe d'observations relatif au couple "éb" imposant de manière trop stricte la position de l'accent qui parfois se promène loin de sa lettre associée.

Il existe une différence majeure entre ces deux graphes d'observations G.3 et G.4. Le premier (G.3) est un assemblage redondant de graphèmes, les graphèmes qui composent toute séquence incluse dans le graphe forment l'image complète. Le second graphe (G.4) est un assemblage non redondant, l'image est la somme de tous les graphèmes qui le composent. Par conséquent, le modèle de Markov caché qui doit donner sa probabilité d'émission doit tenir compte de tous les nœuds du graphe d'observations. Cette configuration n'est pas celle habituellement choisie dans le domaine de la reconnaissance.

2. Annexes : voir paragraphe E.5, page 281

G.2.3 Calcul de $\mathbb{P}\left(G\left(O\right) \mid M\right)$

La formule G.3 pourrait être appliquée en association avec l'algorithme *forward* (E.2.3), la probabilité serait alors le résultat d'une somme de probabilités obtenues pour chaque séquence incluse dans le graphe d'observations. Dans ce cas, l'intérêt de ce dernier est minime si ce n'est une meilleure représentation de la segmentation. Les lignes qui suivent proposent l'adaptation de l'algorithme *forward* à un graphe d'observations.

Soit un modèle de Markov caché à N états défini par sa matrice de transitions A et ses vecteurs d'entrées et de sorties Π, Θ, soit $O = \{O_1, \ldots, O_G\}$ un ensemble d'observations sur lequel est défini un graphe G décrit par sa matrice A^O et ses vecteurs Π^O, Θ^O. On désigne par $q\left(t\right)$ l'état du modèle de Markov caché à l'instant t et $\{1, \ldots, N\}$ l'ensemble de ses états. On désigne également par $O\left(t\right)$ l'observation à l'instant t et par $\overline{O\left(t\right)}$ le passé de cette observation défini par le graphe G. On pose :

$$\forall i \in \{1, \cdots, N\}, \forall t \geqslant 1, \forall h \in \{1, \cdots, G\}, \ \alpha_t\left(i,h\right) = \mathbb{P}\left(q_t = i,\, O\left(t\right) = O_h,\, \overline{O\left(t\right)} \mid M\right) \tag{G.4}$$

Pour $t = 1$, on obtient :

$$\forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\}, \ \alpha_1\left(i,h\right) = \mathbb{P}\left(q_1 = i,\, O\left(1\right) = O_h \mid M\right) = \pi_h^O \, \pi_i \, b_{i,O_h} \tag{G.5}$$

On établit la récurrence suivante sur t :

$$\forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\}, \ \alpha_{t+1}\left(j,h\right) = \sum_{i=1}^{N} \sum_{g=1}^{G} \alpha_t\left(i,g\right) \, a_{ij} \, a_{gh}^O \, b_{j,O_h} \tag{G.6}$$

Enfin, on note T la longueur de la séquence la plus longue, la probabilité du graphe d'observations sachant le modèle :

$$\mathbb{P}\left(G\left(O\right) \mid M\right) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{h=1}^{G} \alpha_t\left(i,h\right) \, \theta_i \, \theta_h^O \tag{G.7}$$

Le résultat (G.7) se démontre de la même manière que celle obtenue en (E.7). Ces formules (G.4) à (G.7)

débouchent sur l'algorithme suivant :

Algorithme G.2.1 : probabilité d'un graphe d'observations
Les notations utilisées sont celles des formules (G.4) à (G.7). T désigne la longueur de la plus grande séquence d'observations incluse dans le graphe G contenant les observations $\{O_1, \ldots, O_G\}$. T vérifie $T \leqslant G$.

Etape A : initialisation

$$\forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\}, \quad \alpha_1(i,o) \longleftarrow \sum_{i=1}^{N} \sum_{g=1}^{G} a_{ij}\, a_{gh}^{O}\, b_{j,O_h}$$

Etape B : récurrence

$$\forall t \in \{1, \ldots, T\}, \forall i \in \{1, \cdots, N\}, \quad \forall h \in \{1, \cdots, G\},$$
$$\alpha_{t+1}(j,h) \longleftarrow \sum_{i=1}^{N} \sum_{g=1}^{G} \alpha_t(i,g)\; a_{ij}\, a_{gh}^{O}\, b_{j,O_h}$$

Etape C : terminaison

$$\mathbb{P}(G(O) \mid M) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{h=1}^{G} \alpha_t(i,h)\; \theta_i\, \theta_h^{O}$$

La suite $\beta'_t(.)$ peut également être adaptée aux graphes d'observations. On désigne également par $O(t)$ l'observation à l'instant t et par $\widehat{O(t)}$ le futur de cette observation défini par le graphe G. On pose :

$$\forall i \in \{1, \cdots, N\}, \forall t \geqslant 1, \forall h \in \{1, \cdots, G\}, \; \beta_t(i,h) = \mathbb{P}\left(O_t = O_h, \widehat{O(t)} \mid q_t = i, M\right) \tag{G.8}$$

Bien que presque équivalente à la définition (E.8), le calcul proposé par les formules (E.9) à (E.11) (page E.9) est inapplicable à la suite (G.8) car le graphe G résume plusieurs séquences de longueurs variables. Le raisonnement qui mène aux formules qui suivent est analogue à celui développé dans [Lallican1999] lors de la réestimation des transitions entre états muets.

Pour $t = T + 1$, on obtient pour :

$$\forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\}, \; \beta'_{T+1}(i,h) = \mathbb{P}(q_{T+1} = i, O_{T+1} = O_h \mid M) = 0 \tag{G.9}$$

On établit la récurrence suivante sur t :

$$\forall t \in \{1, \cdots, T\}, \forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\},$$
$$\beta'_t(i,g) = \sum_{j=1}^{N} \sum_{h=1}^{G} a_{ij}\, a_{gh}^{O}\, b_{j,O_g}\, \beta'_{t+1}(i,g) + b_{i,O_g}\, \theta_i\, \theta_g^{O} \tag{G.10}$$

Enfin, la probabilité du graphe d'observations sachant le modèle :

$$\mathbb{P}(G(O) \mid M) = \sum_{i=1}^{N} \sum_{h=1}^{G} \beta'_1(i,h)\; \pi_i\, \pi_h^{O} \tag{G.11}$$

Le résultat (G.11) se démontre de la même manière que celle obtenue en (E.11). Ces formules (G.8) à (G.11) débouchent sur l'algorithme suivant :

Algorithme G.2.2 : probabilité d'un graphe d'observations
Les notations utilisées sont celles des formules (G.8) à (G.11). T désigne la longueur de la plus grande séquence d'observations incluse dans le graphe G, T vérifie $T \leqslant G$.
Etape A : initialisation
$\forall i \in \{1, \cdots, N\}, \forall h \in \{1, \cdots, G\},$
$\beta'_{T+1}(i,o) = \mathbb{P}(q_{T+1} = i,\, O_{T+1} = O_h \mid M) = 0$
Etape B : récurrence
$\forall t \in \{1, \cdots, T\},\ \forall i \in \{1, \cdots, N\},\ \forall h \in \{1, \cdots, G\},$
$\beta'_t(i,g) = \sum_{j=1}^{N} \sum_{h=1}^{G} a_{ij}\, a^O_{gh}\, b_{j,O_g}\, \beta'_{t+1}(i,g) + b_{i,O_g}\, \theta_i\, \theta^O_g$
Etape C : terminaison
$\mathbb{P}(G(O) \mid M) = \sum_{i=1}^{N} \sum_{h=1}^{G} \beta'_1(i,h)\ \pi_i\, \pi^O_h$

Remarque G.2.3: β ou β'
Pour un graphe G réduit à une séquence d'observations, les suites β (E.8) et β' (G.8) différent d'un facteur :

$$\beta_t(i,h) = \begin{cases} 0 \text{ si } b_{i,h} = 0 \\ \\ \dfrac{\beta'_t(i,h)}{b_{i,h}} \text{ sinon} \end{cases}$$

G.2.4 Apprentissage d'un modèle de Markov caché avec des graphes d'observations

La remarque G.2.3 permet d'utiliser presque telles quelles les formules définies dans la table E.2. On suppose que cette apprentissage est effectué pour les ensembles de d'observations $\left\{O^1, \ldots, O^K\right\}$ et leurs graphes $\left\{G^1, \ldots, G^K\right\}$ correspondant. Les formules de réestimation concernant les probabilités d'émission se déduisent facilement des tables E.2 (émissions discrètes) et E.3 (émissions continues) et sont exprimées dans la table G.1.

G.3 Composition de graphes

Les résultats obtenus dans les paragraphes précédents concernent le calcul de la probabilité d'émission d'un graphe d'observations par une chaîne de Markov cachée. Il s'agit maintenant d'adapter les mécanismes décrits au chapitre 4.5, par conséquent de déterminer la probabilité d'émission d'un graphe d'observations par un modèle constitué d'un graphe de chaînes de Markov cachées. La formalisation utilisée jusqu'à présent rend difficile l'écriture des formules de mise à jour des coefficients. Les modèles de mots construits sont une superposition de modèles de trois graphes : graphe de segmentation pour les graphèmes, graphe d'états pour la chaîne de Markov, graphe de lettres pour les modèles de mots.

Les *Graph Transformer Network* développés dans [Bengio1992] (voir également [Bengio1996]) permettent d'atteindre ce but en construisant un opérateur entre deux graphes permettant d'extraire tous les chemins compatibles (ou communs) des deux graphes.

$$
\begin{aligned}
\overline{\pi_i} &= \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{g=1}^{G} \pi_g^O \, \pi_i \, \beta_1'^k (i,g) \right]}{K} \\
\overline{a_{ij}} &= \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k-1} \sum_{g=1}^{G} \sum_{h=1}^{G} a_{gh}^O \, \alpha_t^k (i,g) \; a_{ij} \, \beta_{t+1}'^k (j,h) \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \sum_{g=1}^{G} \alpha_t^k (i,g) \; \beta_t'^k (i,g) \right]} \\
\overline{\theta_i} &= \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \sum_{g=1}^{G} \theta_g^O \, \alpha_t^k (i,g) \; \theta_i \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \sum_{g=1}^{G} \alpha_t^k (i,g) \; \beta_t'^k (i,g) \right]}
\end{aligned}
$$

Tab. G.1: Formules de réestimation de Baum-Welch.

La figure G.6 reprend la figure illustrée 4.28 (page 110). La forte probabilité du mot "Georges" résulte de la concordance entre l'unique chemin de la segmentation en graphèmes et les multiples chemins inclus dans ce modèle de mot. En quelque sorte, la forte probabilité résulte du fait qu'il existe de nombreux chemins communs entre le graphe de segmentation et le graphe correspond au modèle de mot.

Cette représentation sous forme de graphe relie les modèles de Markov cachés aux réseaux bayésiens dont les probabilités de transitions dépendent de toutes celles qui précèdent. Ces réseaux ne peuvent pas non plus inclure de cycles. Toutefois, à partir du moment où cette contrainte est respectée, les deux modélisations s'expriment de manière identique, aussi bien au niveau du calcul des probabilités que de l'estimation des coefficients.

G.3.1 Définition

L'idée de [Bengio1992] consiste à construire un troisième graphe à partir des deux premiers ne contenant que leur partie commune. Cette opération est appelé une *composition*. On suppose alors qu'un graphe est constitué d'un ensemble de nœuds reliés par des arcs pondérés par une probabilité et contenant des labels :

Définition G.3.1 : graphe orienté
Un graphe est défini par la donnée de l'ensemble N de ses nœuds et de l'ensemble A de ses arcs orientés. Chaque arc $a \in A$ relie les nœuds n_a, n'_a, cet arc est également pondéré par une probabilité $w_a \in [0,1]$ qui représente la probabilité qu'un chemin emprunte cet arc sachant qu'il passe par le nœud n_a. Cet arc a contient aussi un label l_a. Le graphe possède deux nœuds particuliers désignés par le nœud d'entrée e et le nœud de sortie s. Ces nœuds vérifient :

$$\forall a \in A, \; n'_a \neq e \text{ et } n_a \neq s$$

Un graphe est entièrement défini par la donnée de ces nœuds et de ces arcs. On écrit que $G = \{N, A, e, s\}$.

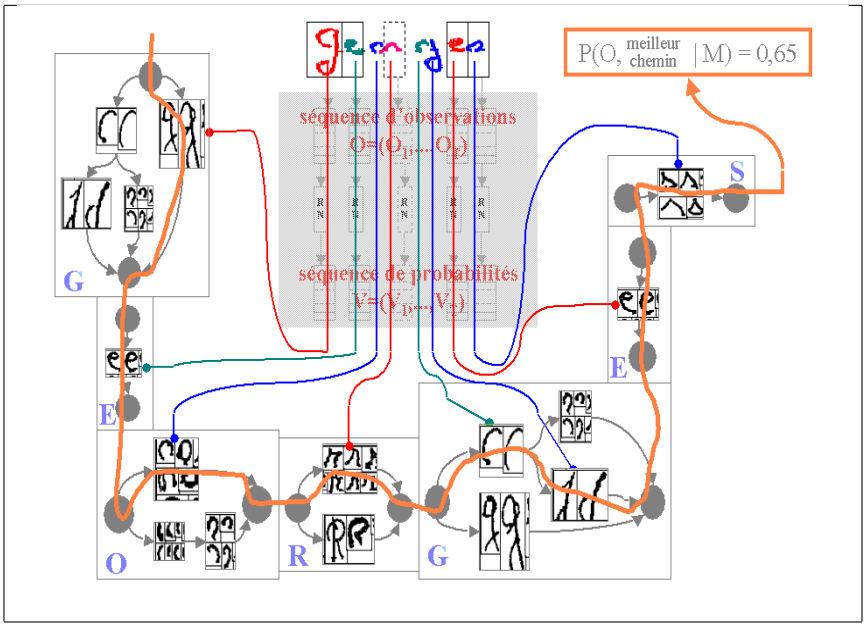


Fig. G.6: Décryptage d'un mot à l'aide d'un modèle hybride réseau de neurones et modèle de Markov caché. Le modèle de Markov caché associé au mot Georges résulte de la juxtaposition des modèles associés aux lettres qui le composent. Chacun d'entre eux est illustré par les séquences de classes graphèmes les plus courantes qui permettent d'écrire une lettre.

G.3.2 Composition

Le graphe de segmentation graphème ainsi qu'une chaîne de Markov sont des graphes orientés vérifiant la définition G.3.1. Il reste à définir la composition entre deux graphes.

Algorithme G.3.2 : composition de graphes orientés
Soient $G_1 = \{N_1, A_1, e_1, s_1\}$ et $G_2 = \{N_2, A_2, e_2, s_2\}$ deux graphes orientés vérifiant la définition G.3.1. On note par $G = \{N, A\} = Comp(G_1, G_2)$ le graphe résultant de la composition des deux premiers graphes. Les ensembles N et A sont construits de manière itérative. Auparavant, on note $n = Comp(n_1, n_2)$ le nœud résultat de la composition des nœuds $n_1 \in N_1$ et $n_2 \in N_2$. De même $a = Comp(a_1, a_2)$ où $a_1 \in A_1$ et $a_2 \in A_2$. a n'existe que si $l_{a_1} = l_{a_2}$ et il vérifie :

$$\begin{aligned} n_a &= Comp(n_{a_1}, n_{a_2}) \\ n'_a &= Comp(n'_{a_1}, n'_{a_2}) \\ w_a &= w_{a_1} w_{a_2} \end{aligned}$$

Etape A : initialisation
$N \longleftarrow e = Comp(e_1, e_2)$
$M \longleftarrow N$
$A \longleftarrow \emptyset$

Etape B : récurrence
Soit $n = Comp(n_1, n_2) \in M$, $M \longleftarrow M - \{n\}$
pour chaque $a_1 \in A_1$ **faire**
  pour chaque $a_2 \in A_2$ **faire**
    si $n_{a_1} = n_1$ **et** $n_{a_2} = n_2$ **et** $l_{a_1} = l_{a_2}$ **alors**
      $A \longleftarrow A \cup \{Comp(a_1, a_2)\}$
      $n' \longleftarrow Comp(n'_{a_1}, n'_{a_2})$
      si $n' \notin N$ **alors**
        $M \longleftarrow M \cup \{n\}$
        $N \longleftarrow N \cup \{n'\}$
      fin si
    fin si
  fin pour
fin pour

Etape C : terminaison
Tant que $M \neq \emptyset$, retour à l'étape B.
Sinon, $Comp(G_1, G_2) = \{A, N, Comp(e_1, e_2), Comp(s_1, s_2)\}$.

Remarque G.3.3: cycles
L'algorithme G.3.2 tel qu'il est décrit ici ne peut être utilisé dans le cas où les deux graphes contiennent des cycles et des chemins communs les incluant. Etant donné que l'algorithme "déroule" les chemins, cette configuration ne permet pas à ce dernier d'aboutir à un résultat fini. La prise en compte des cycles est possible mais inutile si les graphes concernent la reconnaissance de l'écriture manuscrite car le graphe de segmentation en graphèmes ne peut en contenir du fait du sens gauche-droite de l'écriture.

La figure G.9 illustre le résultat de la composition des graphes des figures G.7 et G.8. Chaque contient un label correspondant à une lettre. Les chemins communs aux deux graphes composés définissent la même séquence de symboles. Dans le cadre de la reconnaissance, le label de chaque arc est une classe d'observations tandis que la probabilité associée est le produit d'une probabilité de transition et d'une probabilité d'émission.

Remarque G.3.4: égalité des labels
L'algorithme G.3.2 suppose que le label d'un arc résultant d'une composition est identique aux labels

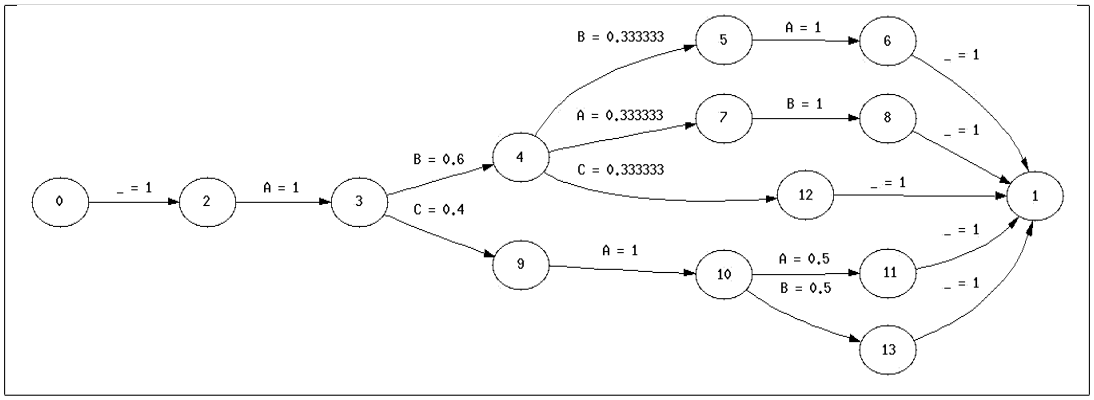


Fig. G.7: Premier graphe, chaque arc est pondéré par une probabilité et son label est une lettre.

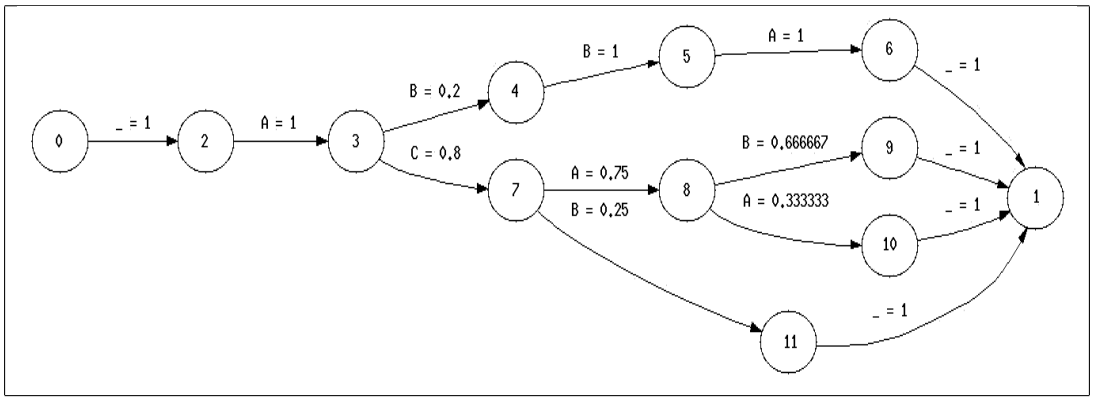


Fig. G.8: Second graphe, chaque arc est pondéré par une probabilité et son label est une lettre. La composition avec le graphe de la figure G.7 va permettre d'extraire l'ensemble des séquences de labels communes aux deux graphes.

des arcs composés. Toutefois, cette contrainte n'est pas nécessaire, l'arc composé peut être quelconque. De même, l'égalité entre deux labels comme condition préalable à la composition de deux arcs peut tout-à-fait être remplacée par une fonction qui détermine si deux arcs peuvent être composés.

Ces algorithmes sont plus souvent reliés aux automates à états finis (Finite State Machine, Weighted Finite Automata) qu'aux graphes (voir [Pereira1997]). Ce formalisme est couramment utilisé en reconnaissance de la parole (voir [Mohri2002a]). Pour de grands graphes, l'élagage de chemins trop peu probables et la composition sont parfois associés afin d'éviter de trop longs temps de calcul.

G.3.3 Calcul de la probabilité du graphe

La composition des deux graphes permet d'extraire toutes les écritures du modèle de reconnaissance compatibles avec la séquence de graphèmes (voir figure G.9). La probabilité de reconnaissance $\mathbb{P}(S \mid M)$ où S est la séquence de graphèmes et M le modèle de reconnaissance s'exprime comme une somme de produits de probabilités :

$$\mathbb{P}(S \mid M) = \sum_{chemin} \mathbb{P}(chemin \mid S)\ \mathbb{P}(chemin \mid M)$$

Cette probabilité s'exprime plus facilement avec le graphe G résultant de la composition de S et M :

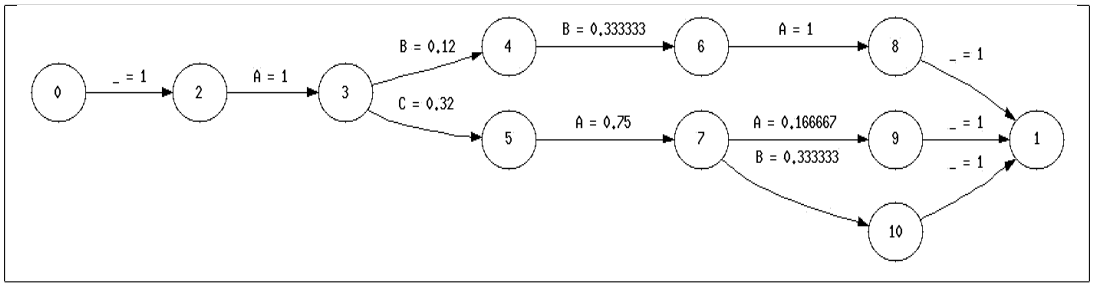


Fig. G.9: Graphe résultant de la composition des graphes illustrés par les figures G.7 et G.8. Il regroupe l'ensemble des chemins communs à ces deux graphes. Les chemins communs sont "ABBA", "ACAA", "ACAB".

$$\mathbb{P}(S \mid M) = \mathbb{P}(G) = \sum_{chemin} \mathbb{P}(chemin \mid G)$$

Cette probabilité se calcule aisément avec l'algorithme *forward* [3] suivant :

Algorithme G.3.5 : algorithme forward
Soit $G = \{N, A, e, s\}$ un graphe orienté. La fonction $h(n \in N)$ est égale au nombre d'arcs arrivant au nœud n : $h(n) = card(a \in A \mid n'_a = n)$. On désigne également par $for(n) \subset A$ l'ensemble des arcs partant de n : $for(n) = \{a \in A \mid n_a = n\}$.

Etape A : initialisation
pour chaque $n \in N$ **faire**
$f(n) \longleftarrow \mathbb{1}_{\{n=e\}}$
$g(n) \longleftarrow 0$
fin pour
$M \longleftarrow \{e\}$

Etape B : propagation
tant que ($M \neq$**) faire**
soit $n \in M$, $M \longleftarrow M - \{n\}$
pour chaque $a \in for(n)$ **faire**
$f(n'_a) \longleftarrow f(n'_a) + f(n)\, w_a$
$g(n'_a) \longleftarrow g(n'_a) + 1$
si $g(n'_a) = h(n'_a)$ **alors**
$M \longleftarrow M \cup \{n'_a\}$
fin si
fin pour
fin tant que

Etape C : terminaison
$P = f(s)$ est la probabilité cherchée.

Cet algorithme propage les probabilités depuis l'entrée jusqu'à la sortie d'où son nom. La version symétrique, l'algorithme *backward*, propage les probabilités depuis la sortie jusqu'à l'entrée. A l'instar de l'algorithme forward qui construit la fonction f, l'algorithme backward construit une fonction b. Ces deux fonctions réunies permettent de calculer le gradient $\frac{\partial P}{\partial w_a}$ de la probabilité P par rapport coefficients de l'arc a :

3. Annexes : voir paragraphe E.2.3, page 250

$$\frac{\partial P}{\partial w_a} = f\left(n_a\right)\ b\left(n'_a\right) \tag{G.12}$$

Au cas où cet arc est la composition de deux arcs a_1 et a_2, il est possible de calculer :

$$\frac{\partial P}{\partial w_{a_1}} = \frac{\partial P}{\partial w_a}\, w_{a_2} \text{ et } \frac{\partial P}{\partial w_{a_2}} = \frac{\partial P}{\partial w_a}\, w_{a_1} \tag{G.13}$$

Ces expressions permettent d'appliquer l'apprentissage de Baum-Welch à un graphe orienté résultant de la composition de deux autres graphes orientés. Par extension, ce formalisme permet aisément d'apprendre des graphes orientés issu de deux compositions successives comme c'est le cas pour une reconnaissance intégrant un graphe de segmentation graphème et un graphe de modèles de Markov cachés associés à des lettres et incluant eux-mêmes un graphe d'états. Le formalisme des graphes orientés permet de construire de manière itérative le gradient de cette modélisation à trois couches de graphes sans avoir à écrire de manière explicite le gradient de la vraisemblance par rapport à ces coefficients éparpillés à chaque étage.

G.3.4 Composition avec des arcs non émetteurs

Les arcs non émetteurs permettent de simplifier de réduire le nombre d'arcs tout en conservant un graphe équivalent. Un arc non émetteur ne change pas la séquence dans laquelle il est inclus. La figure G.10 montre un graphe incluant des arcs non émetteurs équivalent au graphe G.11. Ce dernier contient plus de nœuds et plus d'arcs. L'introduction des arcs non émetteurs permet de réduire la taille des graphes. Toutefois l'algorithme de composition de graphes avec arcs non émetteurs est plus complexe que la version G.3.2.

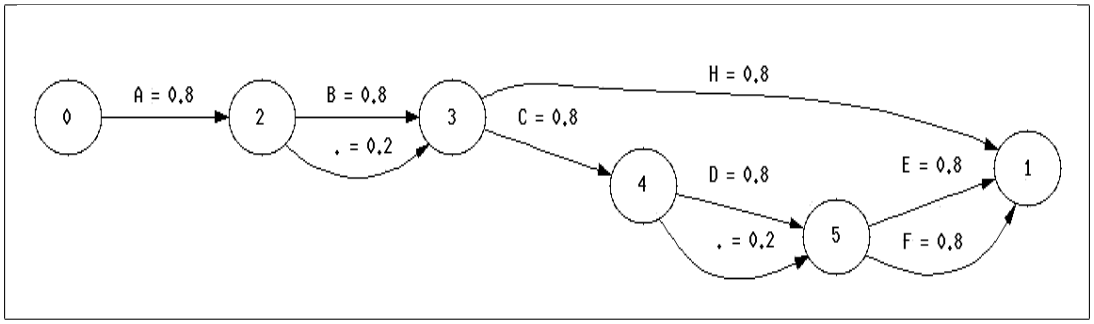


Fig. G.10: Premier graphe à composer, les arcs labellés '.' sont non émetteurs. Le graphe équivalent sans arc non émetteur est illustré par la figure G.11.

L'algorithme de composition de deux graphes contenant des arcs non émetteurs est identique à l'algorithme G.3.6 en ce qui concerne le traitement des nœuds émetteurs. Les nœuds non émetteurs sont traités séparément en introduisant des couples de nœuds supplémentaires dans la liste M. On note $Em(a)$ la propriété émettrice d'un arc a. Si $(n_1, n_2) \in M$, soient deux arcs $a_1 \in for(n_1)$ et $a_2 \in for(n_2)$ tels que $Em(a_1)$ soit vraie et $Em(a_2)$ soit fausse, alors on ajoutera à la liste M la paire de nœud $\left(n_1, n'_{a_2}\right)$ en même temps qu'un nœud non émetteur.

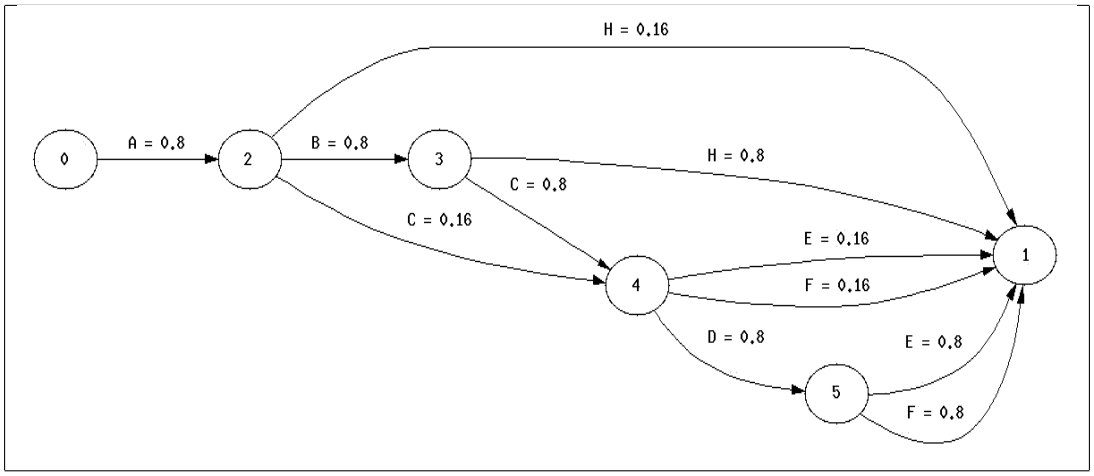


Fig. G.11: Premier graphe à composer, représentation équivalente au graphe G.10 mais sans arc non émetteur. Ce graphe contient plus de nœuds et plus d'arcs.

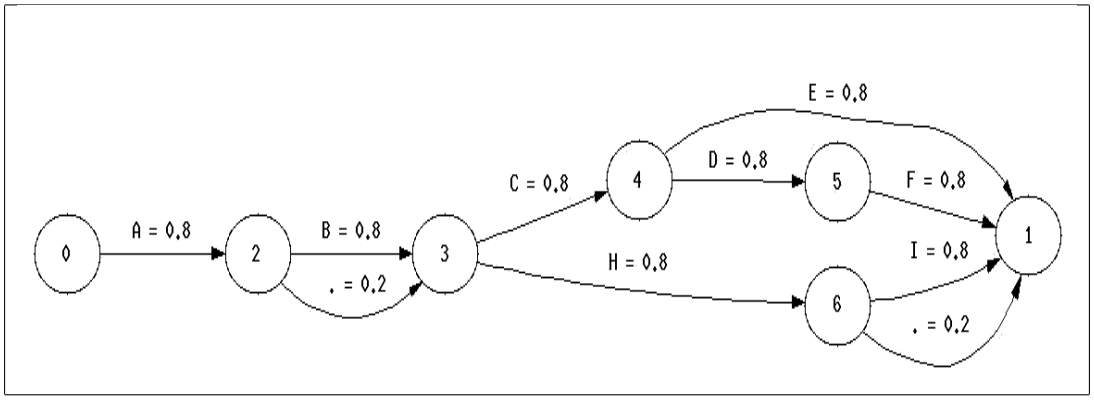


Fig. G.12: Second graphe à composer, les arcs labellés '.' sont non émetteurs.

Algorithme G.3.6 : composition de graphes orientés avec arcs non émetteurs
Soient $G_1 = \{N_1, A_1, e_1, s_1\}$ et $G_2 = \{N_2, A_2, e_2, s_2\}$ deux graphes orientés vérifiant la définition G.3.1. On note par $G = \{N, A\} = Comp\,(G_1, G_2)$ le graphe résultant de la composition des deux premiers graphes. Les ensembles N et A sont construits de manière itérative. Auparavant, on note $n = Comp\,(n_1, n_2)$ le nœud résultat de la composition des nœuds $n_1 \in N_1$ et $n_2 \in N_2$. De même $a = Comp\,(a_1, a_2)$ où $a_1 \in A_1$ et $a_2 \in A_2$. a n'existe que si ($Em(a_1)$ et $Em(a_2)$ et $l_{a_1} = l_{a_2}$) et il vérifie [a] :

$$\begin{array}{rcl} n_a & = & Comp\,(n_{a_1}, n_{a_2}) \\ n'_a & = & Comp\,(n'_{a_1}, n'_{a_2}) \\ w_a & = & \left\{ \begin{array}{ll} w_{a_1} & \text{si } a_2 = \emptyset \\ w_{a_2} & \text{si } a_1 = \emptyset \\ w_{a_1}\, w_{a_2} & \text{sinon} \end{array} \right. \end{array}$$

On supose qu'entre deux nœuds, il n'existe qu'un seul arc non émetteur. L'algorithme utilise deux conditions $Cond_1$ et $Cond_2$ définies en (G.14) et (G.15).

Etape A : initialisation

$N \longleftarrow e = Comp\,(e_1, e_2)$
$M \longleftarrow N$
$A \longleftarrow \emptyset$

a. On adopte comme notation que $n_\emptyset = \emptyset$.

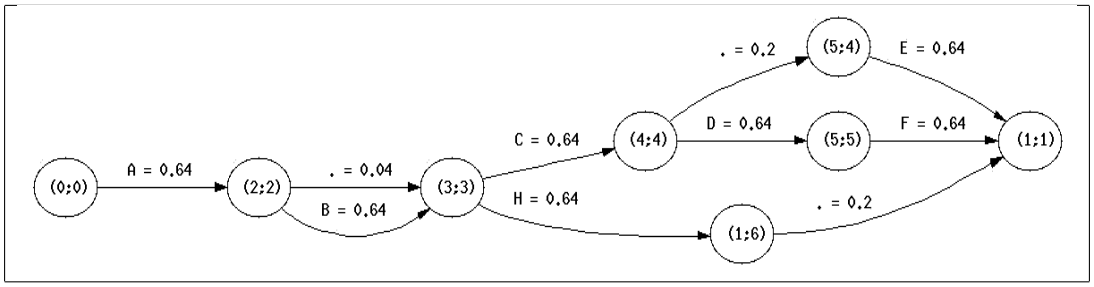


Fig. G.13: Résultat de la composition des graphes des figures G.10, G.12.

Etape B : récurrence

Soit $n = Comp\,(n_1, n_2) \in M$, $M \longleftarrow M - \{n\}$
pour chaque $a_1 \in A_1$ **faire**
pour chaque $a_2 \in A_2$ **faire**
si $n_{a_1} = n_1$ **et** $n_{a_2} = n_2$ **alors**
si non $Em(a_1)$ **et non** $Em(a_2)$ **alors**
$A \longleftarrow A \cup \{Comp\,(a_1, a_2)\}$
$n' \longleftarrow Comp\left(n'_{a_1}, n'_{a_2}\right)$
si $n' \notin N$ **alors**
$M \longleftarrow M \cup \{n\}$
$N \longleftarrow N \cup \{n'\}$
fin si
fin si
si $Em(a_1)$ **et** $Em(a_2)$ **et** $l_{a_1} = l_{a_2}$ **alors**
$A \longleftarrow A \cup \{Comp\,(a_1, a_2)\}$
$n' \longleftarrow Comp\left(n'_{a_1}, n'_{a_2}\right)$
si $n' \notin N$ **alors**
$M \longleftarrow M \cup \{n\}$
$N \longleftarrow N \cup \{n'\}$
fin si
fin si
si $Em(a_1)$ **et non** $Em(a_2)$ **alors**
si $Cond_1\,(N, A, n_1, n_2, a_1, a_2)$ **alors**
$A \longleftarrow A \cup \{Comp\,(\emptyset, a_2)\}$
$n' \longleftarrow Comp\left(n_1, n'_{a_2}\right)$
si $n' \notin N$ **alors**
$M \longleftarrow M \cup \{n\}$
$N \longleftarrow N \cup \{n'\}$
fin si
fin si
fin si
si non $Em(a_1)$ **et** $Em(a_2)$ **alors**
si $Cond_2\,(N, A, n_1, n_2, a_1, a_2)$ **alors**
$A \longleftarrow A \cup \{Comp\,(a_1, \emptyset)\}$
$n' \longleftarrow Comp\left(n'_{a_1}, n_2\right)$
si $n' \notin N$ **alors**
$M \longleftarrow M \cup \{n\}$
$N \longleftarrow N \cup \{n'\}$
fin si
fin si
fin si
fin si
fin pour
fin pour

Etape C : terminaison

Tant que $M \neq \emptyset$, retour à l'étape B.
Sinon, $Comp\,(G_1, G_2) = \{A, N, Comp\,(e_1, e_2)\,, Comp\,(s_1, s_2)\}$.

Les conditions $Cond_1$ et $Cond_2$ permettent d'éviter la création d'un trop grand nombre d'arcs non émetteurs dans le graphe résultant lors que la composition de deux arcs non émetteurs.

$$\begin{array}{c} Cond_1\,(N, A, n_1, n_2, a_1, a_2) \text{ est fausse} \Longleftrightarrow \\ \exists a = Comp\,(\alpha_1, \emptyset) \in A \text{ tel que } \left\{ \begin{array}{l} \text{non } Em(\alpha_1) \text{ et} \\ \exists a^* = Comp\,(\alpha_1, a_2) \in A \end{array} \right. \end{array} \tag{G.14}$$

Par analogie :

$$\begin{array}{c} Cond_2\,(N, A, n_1, n_2, a_1, a_2) \text{ est fausse} \Longleftrightarrow \\ \exists a = Comp\,(\emptyset, \alpha_2) \in A \text{ tel que } \left\{ \begin{array}{l} \text{non } Em(\alpha_2) \text{ et} \\ \exists a^* = Comp\,(a_1, \alpha_2) \in A \end{array} \right. \end{array} \tag{G.15}$$

Si ces tests n'étaient pas effectués, le graphe résultant illustré par la figure G.13 deviendrait celui de la figure G.14.

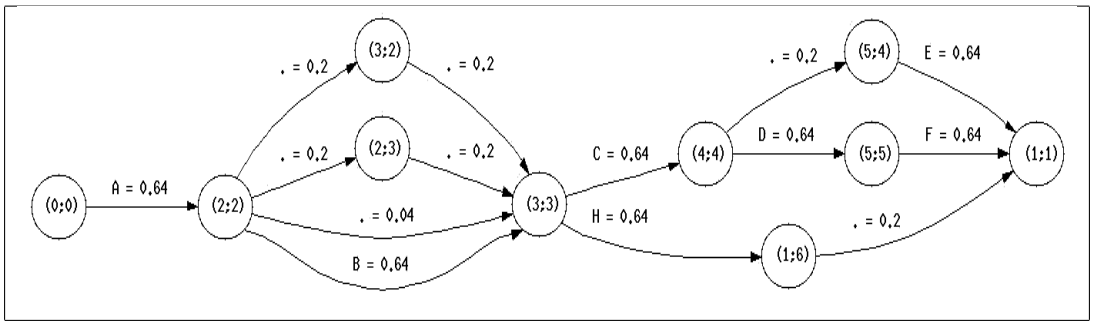


Fig. G.14: Résultat de la composition des graphes des figures G.10, G.12 si les conditions (G.14) et (G.15) étaient toujours vraies dans l'algorithme G.3.6. Entre les nœuds $(2, 2)$ et $(3, 3)$, il existe trois chemins non émetteurs de même probabilité alors que la composition doit retourner un seul arc non émetteur de probabilité 0.04. Les conditions $Cond_1$ (G.14) et $Cond_2$ (G.15) permettent d'éviter la création des nœuds $(3, 2)$ et $(2, 3)$ ainsi que des arcs non émetteurs qui y sont reliés.

Remarque G.3.7: vérification
Pour vérifier la validité de l'algorithme G.3.6, il suffit de comparer le résultat de la composition de deux graphes pour lesquels les arcs non émetteurs ont été enlevés (voir paragraphe G.4.1) avec le résultat de la composition des deux graphes avec arcs non émetteurs, résultat auquel on aura pris soin d'enlever les arcs non émetteurs.

G.4 Algorithmes et graphes

G.4.1 Construction du graphe équivalent sans arc non émetteur

La figure G.15 montre le graphe équivalent à celui de la figure G.12 sans arc non émetteur. A chaque fois qu'un arc non émetteur est rencontré entre les nœuds a et b, tous les arcs partant du nœud b sont dupliqués de telle sorte que les copies partent du nœud a. Les nœuds du graphe sont parcourus de la sortie vers l'entrée pour être certain que tous les arcs successeurs d'un arc non émetteur soient émetteurs. Ceci mène à l'algorithme suivant :

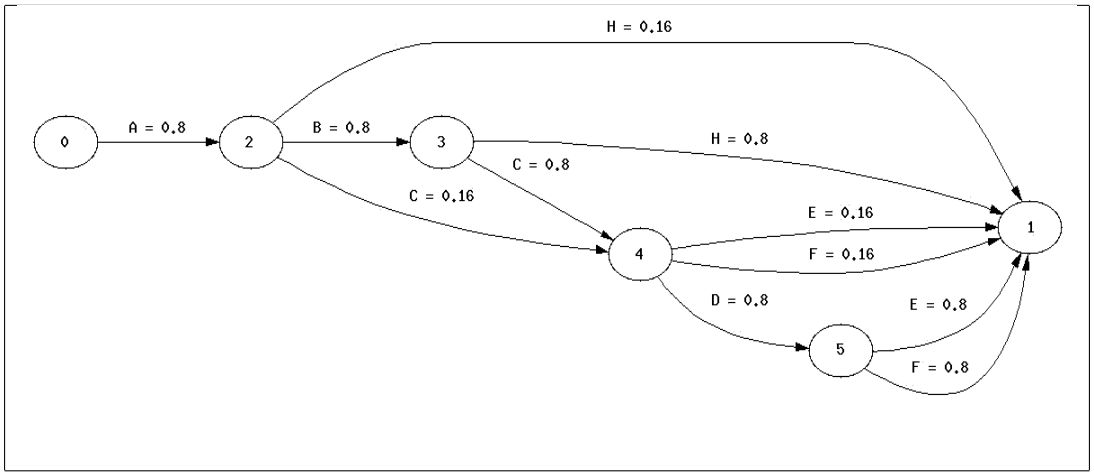


Fig. G.15: Graphe équivalent à celui de la figure G.12 sans arc non émetteur.

Algorithme G.4.1 : graphe sans arc non émetteur
Soit $G = \{N, A, e, s\}$ un graphe orienté vérifiant la définition G.3.1. Pour $n \in N$, on définit la fonction $b(n)$ dénombrant le nombre d'arcs partant du nœud n. La fonction $c(n)$ désigne un compteur. On suppose que ce graphe ne comporte pas de cycles. $S(n)$ désigne l'ensemble des arcs partant du nœud n. La fonction $arc(n_1, n_2, l, w)$ est définie comme étant l'arc reliant les nœuds n_1 et n_2, émettant le symbol l avec la probabilité w.

Etape A : initialisation
$M \longleftarrow \{s\}$ **pour chaque** $n \in N$ **faire**
$b(n) \longleftarrow 0$
$c(n) \longleftarrow 0$
$S(n) \longleftarrow \emptyset$
fin pour
pour chaque $a \in a$ **faire**
$b(n_a) \longleftarrow b(n_a) + 1$
$S(n_a) \longleftarrow S(n_a) \cup \{a\}$
fin pour

Etape B : ajout d'arc non émetteur
tant que $(M \neq \emptyset)$ **faire**
soit $n \in M$, $M \longleftarrow M - \{n\}$
pour chaque $a \in A$ **faire**
$c(n_a) \longleftarrow c(n_a) + 1$
si $Em(a)$ **et** $n'_a = n$ **alors**
pour chaque $\alpha \in S(n'_a)$ **faire**
$A \longleftarrow A \cup \{arc(n_a, n'_\alpha, l_\alpha, w_a\, w_\alpha)\}$
fin pour
$A \longleftarrow A - \{a\}$
fin si
si $c(n_a) = b(n_a)$ **alors**
$c(n_a) \longleftarrow \infty$
$M \longleftarrow M \cup \{n_a\}$
fin si
fin pour
fin tant que

Remarque G.4.2: nœuds ajoutés à la liste M
Seuls les nœuds pour lesquels tous les arcs sortants ont été visités. Ceci garantit que tous les arcs successeurs d'un arc émetteur en train d'être supprimé soient émetteurs.

Cet algorithme, appliqué aux automates à états finis, est aussi connu sous le nom de "ϵ-removal" (voir [Mohri2002b]).

G.4.2 Minimisation

La minimisation d'un graphe est une opération qui consiste à réduire la taille d'un graphe tout en conservant un graphe équivalent. Ces algorithmes dépassent le cadre de ce chapitre. Il est néanmoins possible de se référer à l'article [Mohri2000].

G.4.3 Meilleurs chemins

Pour construire une liste de réponses probabilisés tirés d'un modèle de reconnaissance, il est parfois nécessaire d'extraire les n meilleurs chemins d'un graphe où n n'est pas forcément connu à l'avance. L'algorithme développé dans ce paragraphe propose une optimisation qui prend en compte le fait que les poids associés aux arcs sont positifs, ceci permet l'utilisation d'un algorithme de type Dijkstra ([Dijkstra1971]). Le corps de l'algorithme s'inspire de celui développé dans [Chen1994] (page 490) qui permet d'extraire un à un les meilleurs chemins avec un coût du même ordre de grandeur que le nombre de nœuds dans ces chemins

excepté pour le premier.

Algorithme G.4.3 : meilleurs chemins
Soit $G = \{N, A, e, s\}$ un graphe orienté. On désigne également par $for\,(n) \subset A$ l'ensemble des arcs partant de n : $for\,(n) = \{a \in A \mid n_a = n\}$. On désigne par $b_n\,(t)$ un triplet $(p, a, t) \in [0,1] \times A \times \mathbb{N}^+$ associant :

1. La probabilité du $t^{\text{ième}}$ meilleur chemin allant depuis le premier nœud e au nœud n, on la note $b_n^p\,(t)$.
2. L'arc précédent dans ce chemin depuis l'entrée jusqu'au nœeud n, on le note $b_n^a\,(t)$.
3. L'indice du meilleur chemin arrivant au nœud précédent, on le note $b_n^t\,(t)$.

Par convention, ces trois informations ont pour valeur $\emptyset$ lorsqu'elles n'ont pas encore été calculées. Cette suite $(b_n\,(t))_{t \geqslant 1}$ est constamment triée par ordre de probabilités décroissantes.

Etape A : initialisation
pour chaque $n \in N$ **faire**
$\forall t \geqslant 1,\ b_n \longleftarrow \emptyset$
fin pour
$t \longleftarrow 1$

Etape B : parcourt de l'arbre
$S \longleftarrow \{e\}$
$\forall n \in G,\ f\,(n) \longleftarrow 0$
tant que $(S \neq \emptyset)$ **faire**
Soit $n \in S$, $S \longleftarrow S - \{n\}$
pour chaque $a \in for\,(n)$ **faire**
$f\,(n'_a) \longleftarrow f\,(n'_a) + 1$
$g\,(a) \longleftarrow 1$
si $f\,(n'_a) = card\,(for\,(n'_a))$ **alors**
$S \longleftarrow S \cup \{n'_a\}$
fin si
$b_{n'_a} \longleftarrow b_{n'_a} \overset{\text{insertion}}{\underset{\text{triée}}{\cup}} \left(b_n\,(t)\ w_a, a, t\right)$
fin pour
fin tant que

Etape C : obtention du $t^{\text{ième}}$ meilleur chemin

Soit $(c_i)_i$ le chemin désiré.

$i \longleftarrow 1$

$u_0 \longleftarrow t$

$u_i \longleftarrow b_s^t(t)$

$c_i \longleftarrow b_s^a(t)$

tant que $(n_{c_i} \neq e)$ **faire**

$c_{i+1} \longleftarrow b^a_{n_{c_i}}(u_i)$

$u_{i+1} \longleftarrow b^t_{n_{c_i}}(u_i)$

$i \longleftarrow i+1$

fin tant que

Le $t^{\text{ième}}$ meilleur chemin correspond à la suite d'arcs $(c_{i-k+1})_{1\leqslant k\leqslant i}$ et sa probabilité est $b_s^p(t)$. La suite $(u_{i-k})_{0\leqslant k\leqslant i}$ correspond aux indices de passages à chaque nœud.

Etape D : mise à jour pour l'obtention du meilleur chemin suivant

On parcourt la suite $(c_{i-k+1})_{1\leqslant k\leqslant i}$ pour mettre à jour les informations relatives aux nœuds visités.

pour $k = i$ **à** 1 **faire**

$g(c_k) \longleftarrow g(c_k)+1$

$b_{n'_{c_k}} \longleftarrow b_{n'_{c_k}} \overset{\text{insertion}}{\underset{\text{triée}}{\cup}} \left(b_{n_{c_k}}(g(c_k))\ w_{c_k}, c_k, g(c_k)\right)$

fin pour

Etape E : obtention du meilleur chemin suivant

$t \longleftarrow t+1$

Retour à l'étape C.

Le coût des étapes C et E est de l'ordre du nombre d'arcs du dernier meilleur chemin extrait. Le coût de l'étape B est du même ordre que celui de l'extraction du meilleur chemin avec l'algorithme Dijkstra.

Annexe H

Classification non supervisée

Cette annexe recense différents moyens d'effectuer une classification non supervisée et de déterminer le nombre de classes approprié.

H.1 Algorithme des centres mobiles

H.1.1 Principe

Les centres mobiles ou nuées dynamiques sont un algorithme de classification *non supervisé*. A partir d'un ensemble de points, il détermine pour un nombre de classes fixé, une répartition des points qui minimise un critère appelé *inertie* ou variance *intra-classe*.

Algorithme H.1.1 : centres mobiles
On considère un ensemble de points :

$$(X_i)_{1 \leqslant i \leqslant P} \in \left(\mathbb{R}^N\right)^P$$

A chaque point est associée une classe : $(c_i)_{1 \leqslant i \leqslant P} \in \{1, ..., C\}^P$
On définit les barycentres des classes : $(G_i)_{1 \leqslant i \leqslant C} \in \left(\mathbb{R}^N\right)^C$

Etape A : initialisation
L'initialisation consiste à choisir pour chaque point une classe aléatoirement dans $\{1, ..., C\}$.
$t \longleftarrow 0$

Etape B : calcul des barycentres
pour $k = 1$ **à** C **faire**

$$G_k^t \longleftarrow \frac{\sum_{i=1}^{P} X_i \, \mathbf{1}_{\left\{c_i^t = k\right\}}}{\sum_{i=1}^{P} \mathbf{1}_{\left\{c_i^t = k\right\}}}$$

fin pour
Etape C : calcul de l'inertie

$$I^t \longleftarrow \sum_{i=1}^{P} d^2\left(X_i, G_{c_i^t}^t\right)$$

$$t \longleftarrow t+1$$

si $t > 0$ **et** $I_t \sim I_{t-1}$ **alors**
arrêt de l'algorithme
fin si
Etape D : attribution des classes
pour $i = 1$ **à** P **faire**

$$c_i^{t+1} \longleftarrow \arg\min_k d\left(X_i, G_k^t\right)$$

où $d\left(X_i, G_k^t\right)$ est la distance entre X_i et G_k^t
fin pour
Retour à l'étape B jusqu'à convergence de l'inertie I^t.

Théorème H.1.2 : convergence de l'inertie
Quelque soit l'initialisation choisie, la suite $(I_t)_{t \geqslant 0}$ construite par l'algorithme H.1.1 converge.

La démonstration du théorème H.1.2 nécessite le lemme suivant.

Lemme H.1.3 : inertie minimum
Soit $(X_1, \ldots, X_P) \in \left(\mathbb{R}^N\right)^P$, P points de $\mathbb{R}^N$, le minimum de la quantité $Q\left(Y \in \mathbb{R}^N\right)$:

$$Q(Y) = \sum_{i=1}^{P} d^2(X_i, Y) \tag{H.1}$$

est atteint pour $Y = G = \frac{1}{P}\sum_{i=1}^{P} X_i$ le barycentre des points $(X_1, \ldots, X_P)$.

Démonstration (lemme H.1.3) :

Soit $(X_1, \ldots, X_P) \in \left(\mathbb{R}^N\right)^P$, P points de $\mathbb{R}^N$.

$$\sum_{i=1}^{P}\overrightarrow{GX_i} = \overrightarrow{0} \Longrightarrow \sum_{i=1}^{P} d^2\left(X_i, Y\right) = \sum_{i=1}^{P} d^2\left(X_i, G\right) + P\, d^2\left(G, Y\right)$$
$$\Longrightarrow \underset{Y\in\mathbb{R}^N}{\arg\min} \sum_{i=1}^{P} d^2\left(X_i, Y\right) = \{G\}$$

(H.1.3) □

Démonstration (théorème H.1.1) :

L'étape D consiste à attribuer à chaque point le barycentre le plus proche. On définit J_t par :

$$J^{t+1} = \sum_{i=1}^{P} d^2\left(X_i, G^t_{c_i^{t+1}}\right) \tag{H.2}$$

On en déduit que :

$$\begin{aligned} J^{t+1} &= \sum_{i, c_i^t \neq c_i^{t+1}} d^2\left(X_i, G^t_{c_i^{t+1}}\right) + J^{t+1} \sum_{i, c_i^t = c_i^{t+1}} d^2\left(X_i, G^t_{c_i^{t+1}}\right) \\ J^{t+1} &\leqslant \sum_{i, c_i^t \neq c_i^{t+1}} d^2\left(X_i, G^t_{c_i^t}\right) + \sum_{i, c_i^t = c_i^{t+1}} d^2\left(X_i, G^t_{c_i^t}\right) \\ J^{t+1} &\leqslant I^t \end{aligned} \tag{H.3}$$

Le lemme H.1.3, appliqué à chacune des classes $\{1, \ldots, C\}$, permet d'affirmer que $I^{t+1} \leqslant J^{t+1}$. Par conséquent, la suite $(I_t)_{t\geqslant 0}$ est décroissante et minorée par 0, elle est donc convergente.

(H.1.1) □

Remarque H.1.4: convexité
L'algorithme des centres mobiles cherche à attribuer à chaque point de l'ensemble une classe parmi les C disponibles. La solution trouvée dépend de l'initialisation et n'est pas forcément celle qui minimise l'inertie intra-classe : l'inertie finale est un minimum local. Néanmoins, elle assure que la partition est formée de classes convexes : soit c_1 et c_2 deux classes différentes, on note C_1 et C_2 les enveloppes convexes des points qui constituent ces deux classes, alors $\overset{o}{C_1} \cap \overset{o}{C_2} = \emptyset$. La figure H.1 présente un exemple d'utilisation de l'algorithme des centres mobiles. Des points sont générés aléatoirement dans le plan et répartis en quatre groupes.

H.1.2 Homogénéité des dimensions

Les coordonnées des points $(X_i) \in \mathbb{R}^N$ sont généralement non homogènes : les ordres de grandeurs de chaque dimension sont différents. C'est pourquoi il est conseillé de centrer et normaliser chaque dimension.

On note : $\forall i \in \{1, \cdots, P\},\ X_i = (X_{i,1}, \ldots, X_{i,N})$

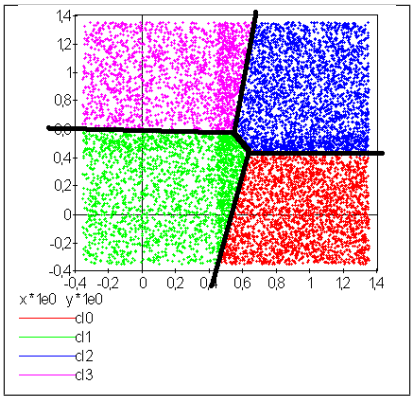


Fig. H.1: Application des centres mobiles : classification en quatre classes d'un ensemble aléatoire de points plus dense sur la partie droite du graphe. Les quatre classes ainsi formées sont convexes.

$$g_k = (EX)_k = \frac{1}{P}\sum_{i=1}^{P} X_{i,k}$$
$$v_{kk} = \left(E\left(X - EX\right)^2\right)_{kk} = \left(EX^2\right)_{kk} - g_k^2$$

Les points centrés et normalisés sont :

$$\forall i \in \{1, \cdots, P\},\ X'_i = \left(\frac{x_{i,1} - g_1}{\sqrt{v_{11}}}, ..., \frac{x_{i,N} - g_N}{\sqrt{v_{NN}}}\right)$$

L'algorithme des centres mobiles est appliqué sur l'ensemble $(X'_i)_{1 \leqslant i \leqslant P}$. Il est possible ensuite de décorréler les variables ou d'utiliser une distance dite de Malahanobis définie par :

$$d_M(X, Y) = X\, M\, Y'$$
où Y' désigne la transposée de Y
et M est une matrice symétrique définie positive

Dans le cas de variables corrélées, la matrice $M = \Sigma^{-1}$ où Σ^{-1} est la matrice de variance-covariance des variables aléatoires $(X_i)_i$.

H.1.3 Estimation de probabilités

A partir de cette classification en C classes, on construit un vecteur de probabilités pour chaque point $(X_i)_{1 \leqslant i \leqslant P}$ en supposant que la loi de X sachant sa classe c_X est une loi normale multidimensionnelle. La classe de X_i est notée c_i. On peut alors écrire :

$$\forall i \in \{1, \cdots, C\},$$

$$G_i = E\left(X \mathbb{1}_{\{c_X=i\}}\right) = \frac{\sum_{k=1}^{P} X_k \mathbb{1}_{\{c_k=i\}}}{\sum_{k=1}^{P} \mathbb{1}_{\{c_k=i\}}}$$

$$V_i = E\left(XX' \mathbb{1}_{\{c_X=i\}}\right) = \frac{\sum_{k=1}^{P} X_k X_k' \mathbb{1}_{\{c_k=i\}}}{\sum_{k=1}^{P} \mathbb{1}_{\{c_k=i\}}}$$

$$\mathbb{P}(c_X = i) = \sum_{k=1}^{P} \mathbb{1}_{\{c_k=i\}} \tag{H.4}$$

$$f(X|c_X = i) = \frac{1}{(2\pi)^{\frac{N}{2}} \sqrt{\det(V_i)}} e^{-\frac{1}{2}(X-G_i)' V_i^{-1} (X-G_i)}$$

$$f(X) = \sum_{k=1}^{P} f(X|c_X = i)\, \mathbb{P}(c_X = i) \tag{H.5}$$

On en déduit que :

$$\mathbb{P}(c_X = i|X) = \frac{f(X|c_X = i)\, \mathbb{P}(c_X = i)}{f(X)} \tag{H.6}$$

La densité des obervations est alors modélisée par une mélange de lois normales, chacune centrée au barycentre de chaque classe. Ces probabilités peuvent également être apprises par un réseau de neurones classifieur [1] où servir d'initialisation à un algorithme EM [2].

H.2 Sélection du nombre de classes

H.2.1 Critère de qualité

L'algorithme H.1.1 effectue une classification non supervisée à condition de connaître au préalable le nombre de classes et cette information est rarement disponible. Une alternative consiste à estimer la pertinence des classifications obtenues pour différents nombres de classes, le nombre de classes optimal est celui qui correspond à la classification la plus pertinente.

Cette pertinence ne peut être estimée de manière unique, elle dépend des hypothèses faites sur les éléments à classer, notamment sur la forme des classes qui peuvent être convexes ou pas, être modélisées par des lois normales multidimensionnelles, à matrice de covariances diagonales, ... Les deux critères qui suivent sont adaptés à l'algorithme H.1.1. Le critère de Davies-Bouldin (voir [Davies1979]), est minimum lorsque le nombre de classes est optimal (voir figure H.2).

1. Annexes : voir paragraphe C.1.5, page 196
2. Annexes : voir paragraphe H.4.1, page 359

$$DB = \frac{1}{C}\sum_{i=1}^{C}\max_{i\neq j}\frac{\sigma_i+\sigma_j}{d\left(C_i,C_j\right)} \tag{H.7}$$

avec
- C nombre de classes
- σ_i écart-type des distances des observations de la classe i
- C_i centre de la classe i

Le critère de Goodman-Kruskal (voir [Goodman1954]) est quant à lui maximum lorsque le nombre de classes est optimal. Il est toutefois plus coûteux à calculer.

$$GK = \frac{S^+ - S^-}{S^+ + S^-} \tag{H.8}$$

avec

$$S^+ = \left\{(q,r,s,t) \mid d\left(q,r\right) < d\left(s,t\right) \text{ avec } \begin{array}{l}(q,r) \text{ sont dans la même classe}\\ (s,t) \text{ sont dans des classes différentes}\end{array}\right\}$$

$$S^- = \left\{(q,r,s,t) \mid d\left(q,r\right) < d\left(s,t\right) \text{ avec } \begin{array}{l}(q,r) \text{ sont dans des classes différentes}\\ (s,t) \text{ sont dans la même classe}\end{array}\right\}$$

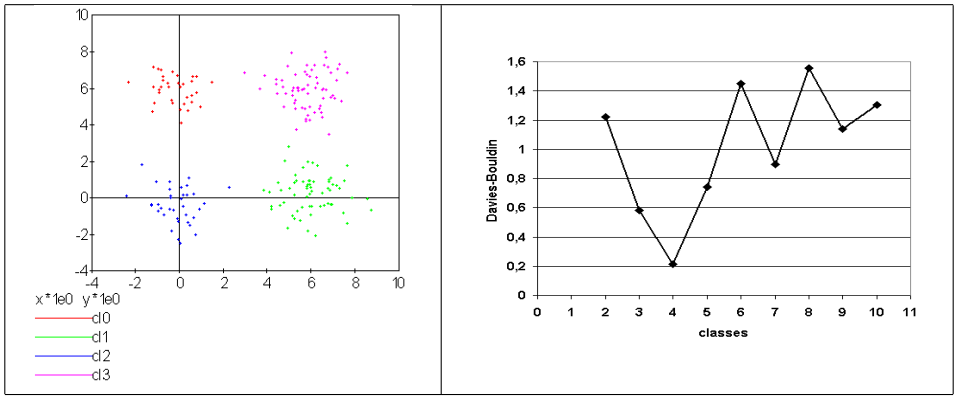


Fig. H.2: Classification en quatre classes : nombre de classes sélectionnées par le critère de Davies-Bouldin (H.7) dont les valeurs sont illustrées par le graphe apposé à droite.

H.2.2 Maxima de la fonction densité

L'article [Herbin2001] propose une méthode différente pour estimer le nombre de classes, il s'agit tout d'abord d'estimer la fonction densité du nuage de points qui est une fonction de $\mathbb{R}^n \longrightarrow \mathbb{R}$. Cette estimation est effectuée au moyen d'une méthode non paramètrique telle que les estimateurs à noyau (voir [Silverman1986] ou paragraphe 4.3.6, page 103). Soit $(X_1,\ldots,X_N)$ un nuage de points inclus dans une image, on cherche à estimer la densité $f_H\left(x\right)$ au pixel x :

$$\hat{f}_H\left(x\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\det H}\,K\left(H^{-1}\left(x-X_i\right)\right) \quad \text{où} \quad K\left(x\right) = \frac{1}{\left(2\pi\right)^{\frac{d}{2}}}\,e^{-\frac{\|x\|^2}{2}} \tag{H.9}$$

H est un paramètre estimée avec la règle de Silverman

L'exemple utilisé dans cet article est un problème de segmentation d'image qui ne peut pas être résolu par la méthode des nuées dynamiques puisque la forme des classes n'est pas convexe, ainsi que le montre la figure H.3. La fonction de densité f est seuillée de manière à obtenir une fonction $g : \mathbb{R}^n \longrightarrow \{0,1\}$ définie par :

$$g(x) = \mathbb{1}_{\{f(x) \geqslant s\}}$$

L'ensemble $g^{-1}(\{1\}) \subset \mathbb{R}^n$ est composée de N composantes connexes notées $(C_1, \ldots, C_N)$, la classe d'un point x est alors l'indice de la composante connexe à la laquelle il appartient ou la plus proche le cas échéant.

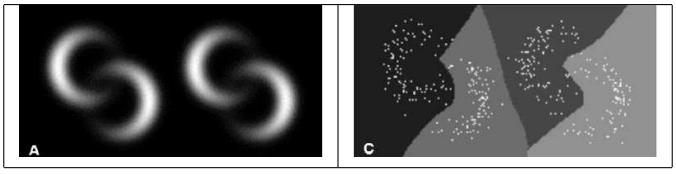


Fig. H.3: Exemple de classification non supervisée appliquée à un problème de segmentation d'image, la première figure montre la densité obtenue, la seconde figure illustre la classification obtenue, figure extraite de [Herbin2001].

Cette méthode paraît néanmoins difficilement applicable lorsque la dimension de l'espace vectoriel atteint de grande valeur. L'exemple de l'image est pratique, elle est déjà découpée en région représentées par les pixels, l'ensemble $g^{-1}(\{1\})$ correspond à l'ensemble des pixels x pour lesquels $f(x) \geqslant s$.

H.2.3 Décroissance du nombre de classes

L'article [Kothari1999] propose une méthode permettant de faire décroître le nombre de classes afin de choisir le nombre approprié. L'algorithme des centres mobiles présentés au paragraphe H.1 proposent de faire décroître l'inertie notée I définie pour un ensemble de points noté $X = (x_1, \ldots, x_N)$ et K classes. La classe d'un élément x est notée $C(x)$. Les centres des classes sont notés $Y = (y_1, \ldots, y_K)$. L'inertie de ce nuage de points est définie par :

$$I = \sum_{x \in X} \left\| x - y_{C(x)} \right\|^2 \tag{H.10}$$

On définit tout d'abord une distance $\alpha \in \mathbb{R}^+$, puis l'ensemble $V(y, \alpha) = \{z \in Y \mid d(y,z) \leqslant \alpha\}$, $V(y,\alpha)$ est donc l'ensemble des voisins des centres dont la distance avec y est inférieur à α. L'article [Kothari1999] propose de minimiser le coût $J(\alpha)$ suivant :

$$J(\alpha) = \sum_{x \in X} \left\| x - y_{C(x)} \right\|^2 + \sum_{x \in X} \sum_{y \in V\left(y_{C(x)}, \alpha\right)} \lambda(y) \left\| y - y_{C(x)} \right\|^2 \tag{H.11}$$

Lorsque α est nul, ce facteur est égal à l'inertie : $I = J(0)$ et ce terme est minimal lorsqu'il y a autant de classes que d'éléments dans X. Lorsque α tend vers l'infini, $J(\alpha) \to J(\infty)$ où :

$$J(\infty) = \sum_{x \in X} \left\| x - y_{C(x)} \right\|^2 + \sum_{x \in X} \sum_{y \in Y} \lambda(y) \left\| y - y_{C(x)} \right\|^2 \tag{H.12}$$

Ici encore, il est possible de montrer que ce terme $J(\infty)$ est minimal lorsqu'il n'existe plus qu'une seule classe. Le principe de cette méthode consiste à faire varier le paramètre α, plus le paramètre α augmente, plus le nombre de classes devra être réduit. Néanmoins, il existe des intervalles pour lequel ce nombre de classes est stable, le véritable nombre de classes de l'ensemble X sera considéré comme celui correspondant au plus grand intervalle stable (voir figure H.4).

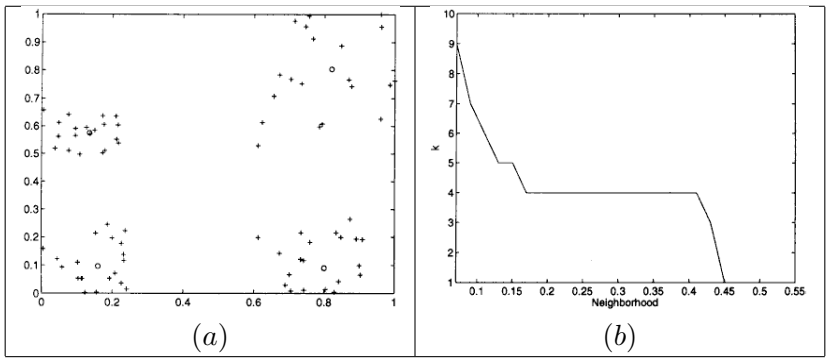


Fig. H.4: Evolutation du nombre de classes en fonction du paramètre α lors de la minimisation du critère $J(\alpha)$, figure extraite de [Kothari1999]. La première image représente le nuage de points illustrant quatre classes sans recouvrement. La seconde image montre que quatre classes est l'état le plus longtemps stable lorsque α croît.

Le coût $J(\alpha)$ est une somme de coût dont l'importance de l'un par rapport à l'autre est contrôle par les paramètres $\lambda(y)$. Le problème de minimisation de $J(\alpha)$ est résolu par l'algorithme H.2.1, il s'appuie sur la méthode des multiplicateurs de Lagrange.

Algorithme H.2.1 : sélection du nombre de classes (Kothari1999)
Les notations sont celles utilisés dans les paragraphes précédents. On suppose que le paramètre α évolue dans l'intervalle $[\alpha_1, \alpha_2]$ à intervalle régulier α_t. Le nombre initial de classes est noté K et il est supposé surestimer le véritable nombre de classes. Soit $\eta \in]0,1[$, ce paramètre doit être choisi de telle sorte que dans l'algorithme qui suit, l'évolution des centres y_k soit autant assurée par le premier de la fonction de coût que par le second.

Etape A : initialisation

$\alpha \longleftarrow \alpha_1$

On tire aléatoirement les centres des K classes $(y_1, \ldots, y_K)$.

Etape B : préparation

On définit les deux suites entières $\left(c_1^1, \ldots, c_K^1\right)$, $\left(c_1^2, \ldots, c_K^2\right)$, et les deux suites de vecteur $\left(z_1^1, \ldots, z_K^1\right)$, $\left(z_1^2, \ldots, z_K^2\right)$.

$$\begin{array}{lll} \forall k, & c_k^1 & = 0 \\ \forall k, & c_k^2 & = 0 \\ \forall k, & z_k^1 & = 0 \\ \forall k, & z_k^2 & = 0 \end{array}$$

Etape C : calcul des mises à jour
pour $i = 1$ **à** N **faire**
Mise à jour d'après le premier terme de la fonction de coût $J(\alpha)$.
$w \longleftarrow \arg\min_{1 \leqslant l \leqslant K} \|x_i - y_l\|^2$
$z_w^1 \longleftarrow z_w^1 + \eta (x_i - y_w)$
$c_w^1 \longleftarrow c_w^1 + 1$
Mise à jour d'après le second terme de la fonction de coût $J(\alpha)$.
pour $v = 1$ **à** K **faire**
si $\|y_v - y_w\| < \alpha$ **alors**
$z_v^2 \longleftarrow z_v^2 - (y_v - y_w)$
$c_v^2 \longleftarrow c_v^2 + 1$
fin si
fin pour
pour $v = 1$ **à** K **faire**
$\lambda_v \longleftarrow \frac{c_v^2 \|z_v^1\|}{c_v^1 \|z_v^2\|}$
$y_v \longleftarrow y_v + z_v^1 + \lambda_v z_v^2$
fin pour
fin pour

Etape D : convergence
Tant que l'étape C n'a pas convergé vers une version stable des centres, y_k, retour à l'étape C. Sinon, tous les couples de classes (i, j) vérifiant $\|y_i - y_j\| > \alpha$ sont fusionnés.
$\alpha \longleftarrow \alpha + \alpha_t$
Si $\alpha \leqslant \alpha 2$, retour à l'étape B.

Etape E : terminaison
Le nombre de classes est celui ayant prévalu pour le plus grand nombre de valeur de α.

H.3 Extension des nuées dynamiques

H.3.1 Classes elliptiques

La version de l'algorithme des nuées dynamique proposée dans l'article [Cheung2003] suppose que les classes ne sont plus de forme circulaire mais suivent une loi normale quelconque. La loi de l'échantillon constituant le nuage de points est de la forme :

$$f(x) = \sum_{i=1}^{N} p_i \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma_i}} exp\left(-\frac{1}{2}(x - \mu_i)' \Sigma_i^{-1} (x - \mu_i)\right) \quad \text{(H.13)}$$
$$\text{avec} : \sum_{i=1}^{N} p_i = 1$$

On définit :

$$G(x, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma}} exp\left(-\frac{1}{2}(x - \mu)' \Sigma^{-1} (x - \mu)\right)$$

L'algorithme qui suit a pour objectif de minimiser la quantité pour un échantillon $(X_1, \ldots, X_K)$:

$$I = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}_{\left\{ i = \underset{1 \leqslant j \leqslant N}{\arg\max} G(X_k, \mu_j, \Sigma_j) \right\}} \ln \left[p_i \, G \left(X_k, \, \mu_i, \, \Sigma_i \right) \right] \tag{H.14}$$

Algorithme H.3.1 : nuées dynamiques généralisées
Les notations sont celles utilisées dans ce paragraphe. Soient η, η_s deux réels tels que $\eta > \eta_s$. La règle préconisée par l'article [Cheung2003] est $\eta_s \sim \frac{\eta}{10}$.

Etape A : initialisation

$t \longleftarrow 0$

Les paramètres $\left\{ p_i^0, \, \mu_i^0, \, \Sigma_i^0 \mid 1 \leqslant i \leqslant N \right\}$ sont initialisés grâce à un algorithme des centres mobiles (H.1.1) ou FSCL (H.3.4). $\forall i$, $p_i^0 = \frac{1}{N}$ et $\beta_i^0 = 0$.

Etape B : récurrence

Soit X_k choisi aléatoirement dans $(X_1, \ldots, X_K)$.

$i = \underset{1 \leqslant i \leqslant N}{\arg\min} \, G \left(X_k, \, \mu_i^t, \, \Sigma_i^t \right)$

pour $i = 1$ **à** N **faire**

$$\begin{array}{rcl} \mu_i^{t+1} & = & \mu_i^t + \eta \left(\Sigma_i^t \right)^{-1} \left(X_k - \mu_i^t \right) \\ \beta_i^{t+1} & = & \beta_i^t + \eta \left(1 - \alpha_i^t \right) \\ \Sigma_i^{t+1} & = & (1 - \eta_s) \, \Sigma_i^t + \eta_s \left(X_k - \mu_i^t \right) \left(X_k - \mu_i^t \right)' \end{array}$$

fin pour

pour $i = 1$ **à** N **faire**

$$p_i^{t+1} = \frac{e^{\beta_i^{t+1}}}{\sum\limits_{j=1}^{N} e^{\beta_j^{t+1}}}$$

fin pour

$t \longleftarrow t + 1$

Etape C : terminaison

Tant que $\underset{1 \leqslant i \leqslant N}{\arg\min} \, G \left(X_k, \, \mu_i^t, \, \Sigma_i^t \right)$ change pour au moins un des points X_k (ou convergence du critère (H.14)), retour à l'étape B.

Remarque H.3.2: mise à jour de Σ^{-1}
L'algorithme précédent propose la mise à jour de Σ_i alors que le calcul de $G(., \mu_i, \Sigma_i)$ implique Σ_i^{-1}, par conséquent, il est préférable de mettre à jour directement la matrice Σ^{-1} :

$$\left(\Sigma_i^{t+1} \right)^{-1} = \frac{\left(\Sigma_i^t \right)^{-1}}{1 - \eta_s} \left[I - \frac{\eta_s \left(X_k - \mu_i^t \right) \left(X_k - \mu_i^t \right)' \left(\Sigma_i^t \right)^{-1}}{1 - \eta_s + \eta_s \left(X_k - \mu_i^t \right)' \left(\Sigma_i^t \right)^{-1} \left(X_k - \mu_i^t \right)} \right]$$

H.3.2 Rival Penalized Competitive Learning (RPCL)

Cet algorithme H.3.3, développé dans [Xu1993], est une variante de celui des centres mobiles H.1.1. Il entreprend à la fois la classification et la sélection du nombre optimal de classes à condition qu'il soit inférieur à une valeur maximale à déterminer au départ de l'algorithme. Un mécanisme permet d'éloigner

les centres des classes peu pertinentes de sorte qu'aucun point ne leur sera affecté.

Algorithme H.3.3 : RPCL
Soient $(X_1, \ldots, X_N)$, N vecteurs à classer en au plus T classes de centres $(C_1, \ldots, C_T)$. Soient deux réels α_r et α_c tels que $0 < \alpha_r \ll \alpha_c < 1$.

Etape A : initialisation
Tirer aléatoirement les centres $(C_1, \ldots, C_T)$.
pour $j = 1$ **à** C **faire**
$n_j^0 \longleftarrow 1$
fin pour

Etape B : calcul de poids
Choisir aléatoirement un point X_i.
pour $j = 1$ **à** C **faire**

$$\gamma_j = \frac{n_j}{\sum_{k=1}^{C} n_k}$$

fin pour
pour $j = 1$ **à** C **faire**

$$u_j = \begin{cases} 1 & \text{si} j \in \arg\min\limits_{k} \left[\gamma_k \, d\left(X_i, C_k\right)\right] \\ -1 & \text{si} j \in \arg\min\limits_{j \neq k} \left[\gamma_k \, d\left(X_i, C_k\right)\right] \\ 0 & \text{sinon} \end{cases}$$

fin pour

Etape C : mise à jour
pour $j = 1$ **à** C **faire**

$$C_j^{t+1} \longleftarrow C_j^t + \begin{cases} \alpha_c \left(X_i - C_j\right) & \text{si } u_j = 1 \\ -\alpha_r \left(X_i - C_j\right) & \text{si } u_j = -1 \\ 0 & \text{sinon} \end{cases}$$

$$n_j^{t+1} \longleftarrow n_j^t + \begin{cases} 1 & \text{si } u_j = 1 \\ 0 & \text{sinon} \end{cases}$$

fin pour
$t \longleftarrow t + 1$

Etape D : terminaison
S'il existe un indice j pour lequel $C_j^{t+1} \neq C_j^t$ alors retourner à l'étape B ou que les centres des classes jugées inutiles ont été repoussés vers l'infini.

Pour chaque point, le centre de la classe la plus proche en est rapproché tandis que le centre de la seconde classe la plus proche en est éloigné mais d'une façon moins importante (condition $\alpha_r \ll \alpha_c$). Après convergence, les centres des classes inutiles ou non pertinentes seront repoussés vers l'infini. Par conséquent, aucun point n'y sera rattaché.

L'algorithme doit être lancé plusieurs fois. L'algorithme RPCL peut terminer sur un résultat comme celui de la figure H.5 où un centre reste coincé entre plusieurs autres. Ce problème est moins important lorsque la dimension de l'espace est plus grande.

H.3.3 RPCL-based local PCA

A l'instar de la méthode décrite au paragraphe H.3.1, l'article [Liu2003] propose une extension de l'algorithme RPCL et suppose que les classes ne sont plus de forme circulaire mais suivent une loi normale quelconque. Cette méthode est utilisée pour la détection de ligne considérées ici comme des lois normales

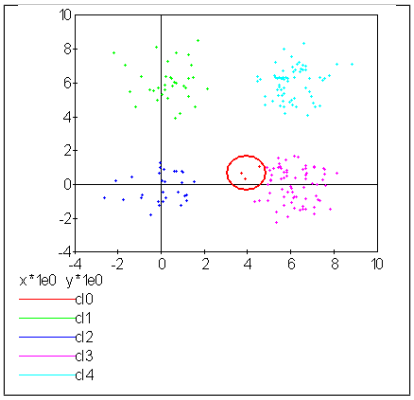


Fig. H.5: Application de l'algorithme RPCL : la classe 0 est incrusté entre les quatre autres et son centre ne peut se "faufiler" vers l'infini.

dégénérées en deux dimensions, la matrice de covariance définit une ellipse dont le grand axe est très supérieur au petit axe, ce que montre la figure H.6. Cette méthode est aussi présentée comme un possible algorithme de squelettisation.

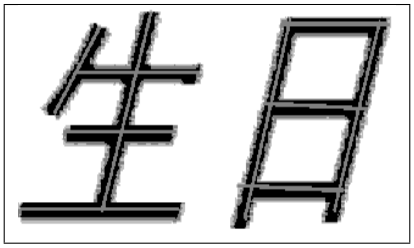


Fig. H.6: Figure extraite de [Liu2003], l'algorithme est utilisé pour la détection de lignes considérées ici comme des lois normales dont la matrice de covariance définit une ellipse dégénérée dont le petit axe est très inférieur au grand axe. Les traits fin grisés correspondent aux classes isolées par l'algorithme RPCL-based local PCA.

On modélise le nuage de points par une mélange de lois normales :

$$f(x) = \sum_{i=1}^{N} p_i \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma_i}} exp\left(-\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1}(x-\mu_i)\right) \quad \text{(H.15)}$$
$$\text{avec : } \sum_{i=1}^{N} p_i = 1$$

On suppose que le nombre de classes initiales N surestime le véritable nombre de classes. L'article [Liu2003] s'intéresse au cas particulier où les matrices de covariances vérifient :

$$\Sigma_i = \zeta_i I + \sigma_i \phi_i \phi_i' \quad \text{(H.16)}$$
$$\text{avec } \zeta_i > 0,\ \sigma_i > 0,\ \phi_i' \phi_i = 1$$

On définit également :

$$G\left(x,\,\mu,\,\Sigma\right)=\frac{1}{\left(2\pi\right)^{\frac{d}{2}}\sqrt{\det\Sigma}}\,exp\left(-\frac{1}{2}\left(x-\mu\right)'\Sigma^{-1}\left(x-\mu\right)\right)$$

L'algorithme utilisé est similaire à l'algortihme RPCL H.3.3. La distance d utilisée lors de l'étape B afin de trouver la classe la plus probable pour un point donné X_k est remplacée par l'expression :

$$d\left(X_k,classe\,i\right)=-\ln p_i^t\,G\left(X_k,\,\mu_i^t,\,\Sigma_i^t\right)$$

L'étape C de mise à jour des coefficients est remplacée par :

$$x^{t+1}\longleftarrow x^t+\left\{\begin{array}{ll}\alpha_c\nabla x^t & \text{si } u_j=1\\ -\alpha_r\nabla x^t & \text{si } u_j=-1\\ 0 & \text{sinon}\end{array}\right.\quad\text{où } x^t \text{ joue le rôle d'un paramètre}$$

Où x^t est remplacé successivement par p_i^t, μ_i^t, ζ_i^t, σ_i^t, ϕ_i^t :

$$\begin{array}{rcl}\nabla p_i^t &=& -\frac{1}{p_i^t}\\ \nabla\mu_i^t &=& -\left(X_k-\mu_i^t\right)\\ \nabla\zeta_i^t &=& \frac{1}{2}\,tr\left[\left(\Sigma_i^t\right)^{-1}\left(I-\left(X_k-\mu_i^t\right)\left(X_k-\mu_i^t\right)'\left(\Sigma_i^t\right)^{-1}\right)\right]\\ \nabla\sigma_i^t &=& \frac{1}{2}\,\left(\phi_i^t\right)'\left(\Sigma_i^t\right)^{-1}\left(I-\left(X_k-\mu_i^t\right)\left(X_k-\mu_i^t\right)'\left(\Sigma_i^t\right)^{-1}\right)\phi_i^t\\ \nabla\phi_i^t &=& \sigma_i^t\left(\Sigma_i^t\right)^{-1}\left(I-\left(X_k-\mu_i^t\right)\left(X_k-\mu_i^t\right)'\left(\Sigma_i^t\right)^{-1}\right)\phi_i^t\end{array}$$

H.3.4 FSCL

L'algorithme Frequency Sensitive Competitive Learning est présenté dans [Balakrishnan1996]. Par rapport à l'algorithme des centres mobiles classique (voir paragraphe H.1), lors de l'estimation des centres des classes, l'algorithme évite la formation de classes sous-représentées.

Algorithme H.3.4 : FSCL
Soit un nuage de points $(X_1,\ldots,X_N)$, soit C vecteurs $(\omega_1,\ldots,\omega_C)$ initialisés de manière aléatoires. Soit $F:(u,t)\in\mathbb{R}^2\longrightarrow\mathbb{R}^+$ croissante par rapport à u. Soit une suite de réels $(u_1,\ldots,u_C)$. Soit une suite $\epsilon\left(t\right)\in[0,1]$ décroissante où t représente le nombre d'itérations. Au début $t\leftarrow 0$.

Etape A : meilleur candidat

Pour un vecteur X_k choisi aléatoirement dans l'ensemble $(X_1,\ldots,X_N)$, on détermine :

$$i^*\in\arg\min\left\{D_i=F\left(u_i,t\right)\,d\left(X_k,\omega_i\right)\right\}$$

Etape B : mise à jour

$\omega_{i^*}\left(t+1\right)\longleftarrow\omega_{i^*}\left(t\right)+\epsilon\left(t\right)\left(X_k-\omega_{i^*}\left(t\right)\right)$
$t\longleftarrow t+1$
$u_{i^*}\longleftarrow u_{i^*}+1$
Retour à l'étape A jusqu'à ce que les nombres $\frac{u_i}{\sum\limits_i u_i}$ convergent.

Exemple de fonctions pour F, ϵ (voir [Balakrishnan1996]) :

$$F(u,t) = u\,\beta e^{-t/T} \text{ avec } \beta = 0,06 \text{ et } 1/T = 0,00005$$
$$\epsilon(t) = \beta\, e^{-\gamma t} \text{ avec } \gamma = 0,05$$

Cet algorithme ressemble à celui des cartes topographiques de Kohonen (voir paragraphe H.4.6) sans toutefois utiliser un maillage entre les neurones (ici les vecteurs ω_i). Contrairement à l'algorithme RPCL, les neurones ne sont pas repoussés s'ils ne sont pas choisis mais la fonction croissante $F(u,t)$ par rapport à u assure que plus un neurone est sélectionné, moins il a de chance de l'être, bien que cet avantage disparaisse au fur et à mesure des itérations.

H.4 D'autres méthodes

H.4.1 Mélange de lois normales

Définition H.4.1 : mélange de lois normales
Soit X une variable aléatoire d'un espace vectoriel de dimension d, X suit un la loi d'un mélange de N lois gaussiennes de paramètres $(\mu_i, \Sigma_i)_{1 \leqslant i \leqslant N}$, alors la densité f de X est de la forme :

$$f(x) = \sum_{i=1}^{N} p_i \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma_i}} exp\left(-\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1}(x-\mu_i)\right) \tag{H.17}$$
$$\text{avec : } \sum_{i=1}^{N} p_i = 1$$

Dans le cas d'une loi normale à valeur réelle $\Sigma = \sigma^2$, l'algorithme permet d'estimer la loi de l'échantillon $(X_1, \ldots, X_T)$, il s'effectue en plusieurs itérations, les paramètres $p_i(0)$, $\mu_i(0)$, $\sigma^2(0)$ sont choisis de manière aléatoire, à l'itération $t+1$, la mise à jour des coefficients est faite comme suit :

$$f_{k,i}(t) = p_i(t) \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma_i(t)}} \exp\left(-\frac{1}{2}(X_k - \mu_i(t))' \Sigma_i^{-1}(t)(X_k - \mu_i(t))\right) \tag{H.18}$$

$$\overline{f_{k,i}}(t) = \frac{f_{k,i}(t)}{\sum\limits_i f_{k,i}(t)} \tag{H.19}$$

$$p_i(t+1) = \frac{1}{T} \sum_{k=1}^{T} \overline{f_{k,i}}(t) \tag{H.20}$$

$$\mu_i(t+1) = \left[\sum_{k=1}^{T} \overline{f_{k,i}}(t)\right]^{-1} \sum_{k=1}^{T} \overline{f_{k,i}}(t) X_k \tag{H.21}$$

$$\Sigma_i^2(t+1) = \left[\sum_{k=1}^{T} \overline{f_{k,i}}(t)\right]^{-1} \sum_{k=1}^{T} \overline{f_{k,i}}(t)\,(X_k - \mu_i(t+1))(X_k - \mu_i(t+1))' \tag{H.22}$$

L'estimation d'une telle densité s'effectue par l'intermédiaire d'un algorithme de type Expectation Maximization (EM) (voir [Dempster1977]) ou de ses variantes SEM, SAEM, ... (voir [Celeux1985],

[Celeux1995]). La sélection du nombre de lois dans le mélange reste un problème ouvert abordé par l'article [Biernacki2001].

H.4.2 Competitive EM algorithm

L'algorithme développé dans l'article [ZhangB2004] tente de corriger les défauts de l'algorithme EM illustrés par la figure H.7. Cette nouvelle version appelée "Competitive EM" ou CEM s'applique à un mélange de lois - normales en particulier -, il détermine le nombre de classes optimal en supprimant ou en ajoutant des classes.

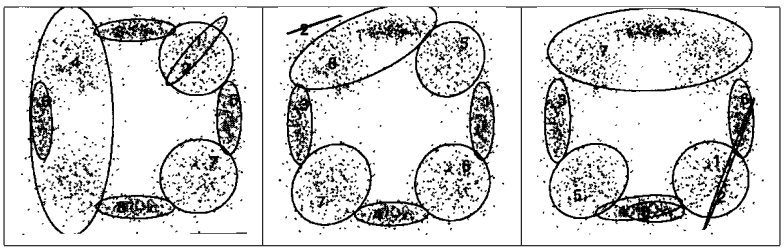


Fig. H.7: Figures extraites de [ZhangB2004], la première image montre deux classes incluant deux autres classes qui devrait donc être supprimées. La seconde image montre une classe aberrante tandis que la troisième image montre des classes se recouvrant partiellement.

On considère un échantillon de variables aléatoires indépendantes et identiquement distribuées à valeur dans un espace vectoriel de dimension d. Soit X une telle variable, on suppose que X suit la loi du mélange suivant :

$$f\left(X \mid \theta\right) = \sum_{i=1}^{k} \alpha_i \, f\left(X \mid \theta_i\right) \tag{H.23}$$

$$\text{avec } \theta = \left(\alpha_i, \theta_i\right)_{1 \leqslant i \leqslant k}, \ \forall i, \ \alpha_i \geqslant 0 \text{ et } \sum_{i=1}^{k} \alpha_i = 1$$

On définit pour une classe m la probabilité $P_{split}(m, \theta)$ qu'elle doive être divisée et celle qu'elle doive être associée à une autre $P_{merge}(m, l, \theta)$. Celles ci sont définies comme suit :

$$P_{split}(m, \theta) = \frac{J\left(m, \theta\right)}{Z\left(\theta\right)} \tag{H.24}$$

$$P_{merge}(m, l, \theta) = \frac{\beta}{J\left(m, \theta\right) Z\left(\theta\right)} \tag{H.25}$$

β est une constante définie par expériences. $J\left(m, \theta\right)$ est défini pour l'échantillon $(x_1, \ldots, x_n)$ par :

$$J\left(m, \theta\right) = \int f_m\left(x, \theta\right) \, \log \frac{f_m\left(x, \theta\right)}{p_m\left(x, \theta_m\right)} \, dx \tag{H.26}$$

$$\text{où } f_m\left(x, \theta\right) = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{x = x_i\}} \, \mathbb{P}\left(m \mid x_i, \theta\right)}{\sum_{i=1}^{n} \mathbb{P}\left(m \mid x_i, \theta\right)}$$

La constante $Z(\theta)$ est choisie de telle sorte que les probabilités $P_{split}(m,\theta)$ et $P_{merge}(m,l,\theta)$ vérifient :

$$\sum_{m=1}^{k} P_{split}(m,\theta) + \sum_{m=1}^{k} \sum_{l=m+1}^{k} P_{merge}(m,l,\theta) = 1 \tag{H.27}$$

L'algorithme EM permet de construire une suite $\hat{\theta}_t$ maximisant la vraisemblance (H.23) à partir de poids $\hat{\theta_0}$. L'algorithme CEM est dérivé de l'algorithme EM [3] [4].

Algorithme H.4.2 : CEM
Les notations sont celles utilisées dans les paragraphes précédents. On suppose que la variable aléatoire $Z = (X, Y)$ où X est la variable observée et Y la variable cachée. T désigne le nombre maximal d'itérations.

Etape A : initialisation

Choix arbitraire de k et $\hat{\theta}_0$.

Etape B : Expectation

$$Q\left(\theta, \hat{\theta}_t\right) = \mathbb{E}\left(\log\left[f\left(X, Y \mid \theta\right)\right] \mid X, \hat{\theta}_t\right) \tag{H.28}$$

Etape C : Maximization

$$\hat{\theta}_{t+1} = \arg\max_{\theta} Q\left(\theta, \hat{\theta}_t\right) \tag{H.29}$$

Etape D : convergence

$t \longleftarrow t+1$

Si $\hat{\theta}_t$ n'a pas convergé vers un maximum local, alors on retourne à l'étape B.

Etape E : division ou regroupement

Dans le cas contraire, on estime les probabilités $P_{split}(m,\theta)$ et $P_{merge}(m,l,\theta)$ définie par les expressions (H.24) et (H.25). On choisit aléatoirement une division ou un regroupement (les choix les plus probables ayant le plus de chance d'être sélectionnés). Ceci mène au paramètre θ'_t dont la partie modifiée par rapport à $\hat{\theta}_t$ est déterminée de manière aléatoire. L'algorithme EM est alors appliqué aux paramètres θ'_t jusqu'à convergence aux paramètres θ''_t.

Etape F : acceptation

On calcule le facteur suivant :

$$P_a = \min\left\{\exp\left[\frac{L\left(\theta''_t, X\right) - L\left(\theta_t, X\right)}{\gamma}\right], 1\right\} \tag{H.30}$$

On génére aléatoirement une variable $u \sim U[0,1]$, si $u \leqslant P_a$, alors les paramètres θ''_t sont validés. $\hat{\theta}_t \longleftarrow \theta''_t$ et retour à l'étape B. Dans le cas contraire, les paramètres θ''_t sont refusés et retour à l'étape E.

3. voir algorithme E.4.13
4. Annexes : voir paragraphe E.4.6, page 276

Etape G : terminaison
Si $t < T$, on retoure à l'étape B. Sinon, on choisit les paramètres $\theta^* = \hat{\theta}_{t^*}$ qui maximisent l'expression :

$$L(\theta^* \mid X) = \log f(X \mid \theta) - \frac{N^*}{2} \sum_{i=1}^{k^*} \log \frac{n\alpha_i^*}{12} - \frac{k^*}{2} \log \frac{n}{12} - \frac{k^*(N^*+1)}{2} \qquad \text{(H.31)}$$

$$\text{avec} \begin{cases} n \text{ est le nombre d'exemples} \\ N \text{ est le nombre de paramètres spécifiant chaque composant} \end{cases}$$

L'article [ZhangB2004] prend $\gamma = 10$ mais ne précise pas de valeur pour β qui dépend du problème. Toutefois, il existe un cas supplémentaire où la classe m doit être supprimée afin d'éviter sa convergence vers les extrêmes du nuage de points à modéliser. Si $n\alpha_m < N$, le nombre moyen de points inclus dans une classe est inférieur au nombre de paramètres attribués à cette classe qui est alors supprimée. Cette condition comme l'ensemble de l'article s'inspire de l'article [Figueiredo2002] dont est tiré le critère décrit en (H.31).

H.4.3 Neural gas

Cette méthode proposée dans [Martinetz1993] constitue une méthode non supervisée de quantification vectorielle (learning vector quantization, LVQ). Toutefois, elle peut aussi être considérée comme une extension de la méthode RPCL vue au paragraphe H.3.2. L'article [Camastra2003] l'applique dans le cadre de reconnaissance caractère et le compare aux différents algorithmes LVQ (1,2,3) et aux cartes de Kohonen (voir paragraphe H.4.6).

Algorithme H.4.3 : Neural Gas
Soient $(X_1, \ldots, X_N)$, N vecteurs à classer et T classes de centres $(C_1, \ldots, C_T)$. Soient quatre réels ϵ_i, ϵ_f, λ_i, λ_f et un nombre d'itérations maximum t_f (des valeurs pratiques pour ces paramètres sont données dans [Martinetz1993]).
Etape A : initialisation
Tirer aléatoirement les centres $(C_1, \ldots, C_T)$.

Etape B : mise à jour
Choisir aléatoirement un point X_i.
Classer les centres C_k par proximité croissante de X_i de sorte que : $d\left(X_i, C_{\sigma(1)}\right) \leqslant ... \leqslant d\left(X_i, C_{\sigma(T)}\right)$
pour $j = 1$ **à** C **faire**

$$C_j^{t+1} \longleftarrow C_j^t + \epsilon_j \left(\frac{\epsilon_f}{\epsilon_j}\right)^{\frac{t}{t_f}} exp\left(-\left[\sigma(j) - 1\right]\left[\lambda_j \left(\frac{\lambda_f}{\lambda_j}\right)^{\frac{t}{t_f}}\right]^{-1}\right)\left(X_i - C_j^t\right)$$

fin pour
$t \longleftarrow t + 1$
Etape C : terminaison
si $t < t_f$ alors retour à l'étape B

Cet algorithme ressemble à celui des cartes de Kohonen (paragraphe H.4.6) sans toutefois imposer de topologie entre les différentes classes. Il ressemble également à l'algorithme RPCL (H.3.3) à ceci près

que lorsqu'un point X_i est choisi aléatoirement, tous les centres des classes sont rapprochés à des degrés différents alors que l'algorithme RPCL rapproche le centre le plus proche et repousse le second centre le plus proche.

H.4.4 Classification ascendante hiérarchique

Comme l'algorithme des centres mobiles (H.1.1), cet algorithme permet également d'effectuer une classification non supervisée des données. Soit un ensemble $E = (x_1, \ldots, x_N)$ à classer, on suppose également qu'il existe une distance entre ces éléments notée $d(x, y)$. De cette distance, on en déduit un critère ou une inertie entre deux parties ne possédant pas d'intersection commune. Par exemple, soient deux parties non vide A et B de E telles que $A \cap B = \emptyset$, on note $|A|$ le nombre d'éléments de A. Voici divers critères possibles :

$$\text{le diamètre } D(A,B) = \max\{d(x,y) \mid x,y \in A \cup B\}$$

$$\text{l'inertie } I(A,B) = \frac{1}{|A \cup B|} \sum_{x \in A \cup B} d(x, G_{A \cup B})$$

où $G_{A \cup B}$ est le barycentre de la partie $A \cup B$

On note $C(A,B)$ le critère de proximité entre deux parties, la classification ascendante hiérarchique consiste à regrouper d'abord les deux parties minimisant le critère $C(A,B)$.

Algorithme H.4.4 : CAH
Les notations sont celles utilisées dans les paragraphes précédents. Soit l'ensemble des singletons $P = (\{x_1\}, \ldots, \{x_N\})$.
Etape A : initialisation
$t \longrightarrow 0$

Etape B : choix des deux meilleures parties
Soit le couple de parties (A,B) défini par :

$$C(A,B) = \min\{C(M,N) \mid M,N \in P, \text{ et } M \neq N\}$$

Etape C : mise à jour
$c_t \longleftarrow C(A,B)$
$P \longleftarrow P - \{A\} - \{B\}$ Tant que $P \neq \{E\}$, $t \longleftarrow t+1$ et retour à
$P \longleftarrow P \cup \{A \cup B\}$
l'étape B.

L'évolution de l'ensemble des parties P est souvent représentée par un graphe comme celui de la figure H.8. C'est ce graphe qui permet de déterminer le nombre de classes approprié à l'ensemble E par l'intermédiaire de la courbe (t, c_t). Le bon nombre de classe est souvent situé au niveau d'un changement de pente ou d'un point d'inflexion de cette courbe. Cette méthode est décrite de manière plus complète dans [Saporta1990].

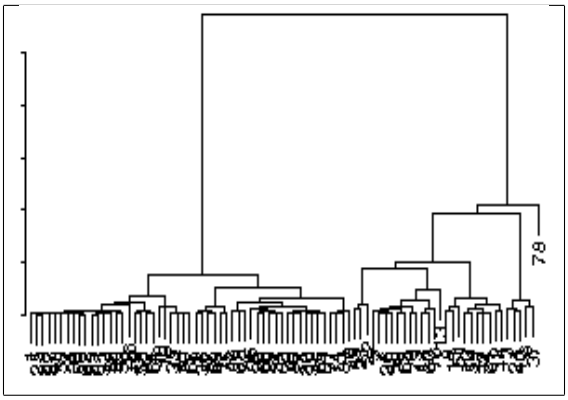


Fig. H.8: Représentation classique de l'arbre obtenu par une CAH. Chaque palier indique un regroupement de deux parties et la valeur du critère de proximité correspondant.

H.4.5 Carte de Kohonen

Les cartes de Kohonen [5] (voir [Kohonen1997]) sont assimilées à des méthodes neuronales. Ces cartes sont constituées d'un ensemble de neurones $(\mu_1, \ldots, \mu_N)$ lesquels sont reliés par une forme récurrente de voisinage (voir figure H.9). Les neurones sont initialement répartis selon ce système de voisinage. Le réseau évolue ensuite puisque chaque point de l'espace parmi l'ensemble $(X_1, \ldots, X_K)$ attire le neurone le plus proche vers lui, ce neurone attirant à son tour ses voisins. Cette procédure est réitérée jusqu'à convergence du réseau en faisant décroître l'attirance des neurones vers les points du nuage.

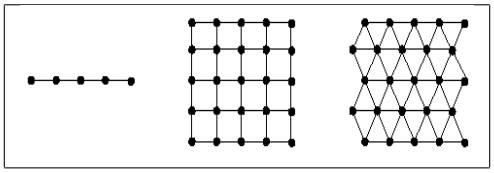


Fig. H.9: Trois types de voisinages couramment utilisés pour les cartes de Kohonen, voisinages linéaire, rectangulaire, triangulaire.

5. Self Organizing Map (SOM) est le terme anglais pour les cartes de Kohonen.

Algorithme H.4.5 : cartes de Kohonen (SOM)
Soient $\left(\mu_1^t, \ldots, \mu_N^t\right) \in \left(\mathbb{R}^n\right)^N$ des neurones de l'espace vectoriel $\mathbb{R}^n$. On désigne par $V\left(\mu_j\right)$ l'ensemble des neurones voisins de μ_j pour cette carte de Kohoen. Par définition, on a $\mu_j \in V\left(\mu_j\right)$. Soit $(X_1, \ldots, X_K) \in \left(\mathbb{R}^n\right)^K$ un nuage de points. On utilise une suite de réels positifs (α_t) vérifiant :

$$\sum_{t \geqslant 0} \alpha_t^2 < \infty \text{ et } \sum_{t \geqslant 0} \alpha_t = \infty$$

Etape A : initialisation
Les neurones $\left(\mu_1^0, \ldots, \mu_N^0\right)$ sont répartis dans l'espace $\mathbb{R}^n$ de manière régulière selon la forme de leur voisinage (voir figure H.9).
$t \longleftarrow 0$

Etape B : neurone le plus proche
On choisi aléatoirement un points du nuage X_i puis on définit le neurone $\mu_{k^*}^t$ de telle sorte que :
$\left\|\mu_{k^*}^t - X_i\right\| = \min\limits_{1 \leqslant j \leqslant N} \left\|\mu_j^t - X_i\right\|$

Etape C : mise à jour
pour chaque $\mu_j^t \in V\left(\mu_{k^*}^t\right)$ **faire**
$\mu_j^{t+1} \longleftarrow \mu_j^t + \alpha_t \left(X_i - \mu_j^{t+1}\right)$
fin pour $t \longleftarrow t+1$
Tant que l'algorithme n'a pas convergé, retour à l'étape B.

L'étape C de mise à jour peut être modifiée de manière à améliorer la vitesse de convergence (voir [Lo1991]) :

$$\mu_j^{t+1} \longleftarrow \mu_j^t + \alpha_t\, h\left(\mu_j^t, \mu_{k^*}^t\right)\, \mu_k \left(X_i - \mu_j^{t+1}\right) \tag{H.32}$$

Où h est une fonction à valeur dans l'intervalle $[0,1]$ qui vaut 1 lorsque $\mu_j^t = \mu_{k^*}^t$ et qui décroît lorsque la distance entre ces deux neurones augmente. Une fonction typique est : $h\left(x,y\right) = h_0 \exp\left(-\frac{\|x-y\|^2}{2\,\sigma_t^2}\right)$.

Les cartes de Kohonen sont utilisées en analyse des données afin de projeter un nuage de points dans un espace à deux dimensions d'une manière non linéaire en utilisant un voisinage rectangulaire. Elles permettent également d'effectuer une classification non supervisée en regroupant les neurones là où les points sont concentrés. Les arêtes reliant les neurones ou sommets de la cartes de Kohonen sont soit rétrécies pour signifier que deux neurones sont voisins, soit distendues pour indiquer une séparation entre classes.

H.4.6 Carte de Kohonen et classification

L'article [Wu2004] aborde le problème d'une classification à partir du résultat obtenu depuis une carte de Kohonen (voir algorithme H.4.5). Plutôt que de classer les points, ce sont les neurones qui seront classés en C classes. Après avoir appliqué l'algorithme de Kohonen (algorithme H.4.5), la méthode proposée dans [Wu2004] consiste à classer de manière non supervisée les A neurones obtenus $(\mu_1, \ldots, \mu_A)$. Toutefois, ceux-ci ne sont pas tous pris en compte afin d'éviter les points aberrants. On suppose que $\alpha_{il} = 1$ si le neurone l est le plus proche du point X_i, 0 dans le cas contraire. Puis on construit les quantités suivantes :

$$\nu_k = \sum_{i=1}^{N} \alpha_{ik} \text{ ainsi que } T_k = \frac{1}{\nu_k} \sum_{i=1}^{N} \alpha_{ik} X_i \text{ et } \theta(T_k) = \sqrt{\frac{1}{\nu_k} \sum_{i=1}^{N} \alpha_{ik} \|X_i - T_k\|^2}$$

$$\text{De plus, } \overline{\theta} = \frac{1}{A} \sum_{k=1}^{A} \theta(T_k) \text{ et } \sigma(\theta) = \sqrt{\frac{1}{A} \sum_{k=1}^{A} \left(\theta(T_k) - \overline{\theta}\right)^2}$$

Si $\nu_k = 0$ ou $\|\mu_k - T_k\| > \overline{\theta} + \sigma(\theta)$, le neurone μ_k n'est pas prise en compte lors de la classification non supervisée. Une fois celle-ci terminée, chaque élément X_i est classé selon la classe du neurone le plus proche.

L'article [Wu2004] propose également un critère permettant de déterminer le nombre de classes idéale. On note, $a_{ik} = 1$ si X_i appartient à la classe k, dans le cas contraire, $a_{ik} = 0$. On définit n_k le nombre d'éléments de la classe k, le vecteur moyenne M_k associé à la classe k :

$$n_k = \sum_{i=1}^{N} a_{ik} \text{ ainsi que } M_k = \frac{1}{n_k} \sum_{i=1}^{N} a_{ik} X_i \text{ et } \sigma^2(M_k) = \frac{1}{n_k} \sum_{i=1}^{N} a_{ik} \|X_i - M_k\|^2$$

On note au préalable $\sigma = \sqrt{\frac{1}{C} \sum_{k=1}^{C} \sigma^2(M_k)}$. L'article définit ensuite la densité interne pour C classes :

$$D_{int}(C) = \frac{1}{C} \sum_{k=1}^{C} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ik} a_{jk} \mathbb{1}_{\{\|X_i - X_j\| \leqslant \sigma\}}$$

On définit la distance d^*_{kl} pour $(k,l) \in \{1,\ldots,C\}^2$, cette distance est égale à la distance minimale pour un couple de points, le premier appartenant à la classe i, le second à la classe j :

$$d^*_{kl} = \min \{\|X_i - X_j\| \mid a_{ik} a_{jl} = 1\} = \left\| X^{kl}_{i^*} - X^{kl}_{j^*} \right\|$$

La densité externe est alors définie en fonction du nombre de classes C par :

$$D_{ext}(C) = \sum_{k=1}^{C} \sum_{l=1}^{C} \left[\frac{d_{kl}}{\sigma(k)\,\sigma(l)} \sum_{i=1}^{N} \mathbb{1}_{\{a_{ik}+a_{il}>0\}} \mathbb{1}_{\left\{ \left\| X_i - \frac{X^{kl}_{i^*} + X^{kl}_{j^*}}{2} \right\| \leqslant \frac{\sigma(k)+\sigma(l)}{2} \right\}} \right]$$

L'article définit ensuite la séparabilité en fonction du nombre de classes C :

$$Sep(C) = \frac{1}{D_{ext}(C)} \sum_{k=1}^{C} \sum_{l=1}^{C} d^*_{kl}$$

Enfin, le critère (H.33) noté [6] $CDBw(C)$ est défini par :

6. Composing Density Between and With clusters

$$CDBw(C) = D_{int}(C) * Sep(C) \tag{H.33}$$

Ce critère est maximal pour un nombre de classes optimal. Outre les résultats de l'article [Wu2004] sommairement résumés ici, ce dernier revient sur l'histoire des cartes de Kohonen, depuis leur création ([Kohonen1982]) jusqu'aux derniers développements récents.

H.4.7 Classification à partir de graphes

L'article [Bandyopadhyay2004] propose une méthode qui s'appuie sur les graphes et permettant de classer automatiquement un nuage de points organisé sous forme de graphe. Chaque élément est d'abord relié à ses plus proches voisins, les arcs du graphe obtenus sont pondérés par la distance reliant les éléments associés chacun à un nœud. Les arêtes sont ensuite classées par ordre croissant afin de déterminer un seuil au delà duquel ces arcs relient deux éléments appartenant à deux classes différentes. Ceci mène à l'algorithme H.4.6. La figure H.10 illustre quelques résultats obtenus sur des nuages de points difficiles à segmenter par des méthodes apparentées aux nuées dynamiques.

Algorithme H.4.6 : classification par graphe de voisinage
On désigne par e_{ij} les arcs du graphe $G(S, A)$ reliant les éléments i et j et pondérés par $d_{ij} = d(x_i, x_j)$ la distance entre les éléments x_i et x_j de l'ensemble $(x_1, \ldots, x_N)$. S désigne l'ensemble des sommets et A l'ensemble des arcs $A = (e_{ij})_{ij}$. On numérote les arêtes de 1 à N^2 de telle sorte qu'elles soient triées : $w_{\sigma(1)} \leqslant w_{\sigma(2)} \leqslant ... \leqslant w_{\sigma(N^2)}$. On élimine dans cette liste les arcs de même poids, on construit donc la fonction σ' de telle sorte que : $w_{\sigma'(1)} < w_{\sigma'(2)} < ... < w_{\sigma'(n)}$ avec $n \leqslant N^2$. On pose $\lambda = 2$.

Etape A : détermination de l'ensemble des arcs à conserver
On désigne par X l'ensemble des arcs à conserver. $X = A$. Si $w_{\sigma'(n)} < \lambda w_{\sigma'(1)}$ alors X est inchangé et on passe à l'étape suivante. Sinon, on construit la suite $\delta_i = w_{\sigma'(i+1)} - w_{\sigma'(i)}$ pour $i \in \{1, \ldots, n-1\}$. La suite $\delta_{\phi(i)}$ correspond à la même suite triée : $\delta_{\phi(1)} \leqslant ... \leqslant \delta_{\phi(n-1)}$. On définit $t = \frac{\delta_{\phi(1)} + \delta_{\phi(n-1)}}{2}$. On définit alors le seuil α tel que :

$$\alpha = \min \left\{ w_{\sigma(i)} \mid 1 \leqslant i \leqslant n-1 \text{ et } w_{\sigma'(i+1)} - w_{\sigma'(i)} \geqslant t \text{ et } w_{\sigma'(i)} \geqslant \lambda w_{\sigma'(1)} \right\}$$

Si α n'est pas défini, X est inchangé et on passe à l'étape suivante, sinon :

$$X = \{ e_{ij} \in A \mid d_{ij} \leqslant \alpha \}$$

Etape B : détermination des classes
Si $X = A$ alors l'algorithme ne retourne qu'une seule classe. Dans le cas contraire, on extrait du graphe $G(S, X)$ l'ensemble des composantes connexes $\{C_1, \ldots, C_p\}$ où p désigne le nombre de composantes connexes du graphe. Si $p > \sqrt{card(X)}$, l'algorithme mène à une sur-segmentation, on ne retourne à nouveau qu'une seule classe. Dans le cas contraire, on applique ce même algorithme à chacune des composantes connexes (C_k) extraites du graphe.

L'algorithme est donc appliqué de manière récursive tant qu'un sous-ensemble peut être segmenté.

L'algorithme H.4.6, puisqu'il est appliqué récursivement, permet de construire une hiérarchie de classes

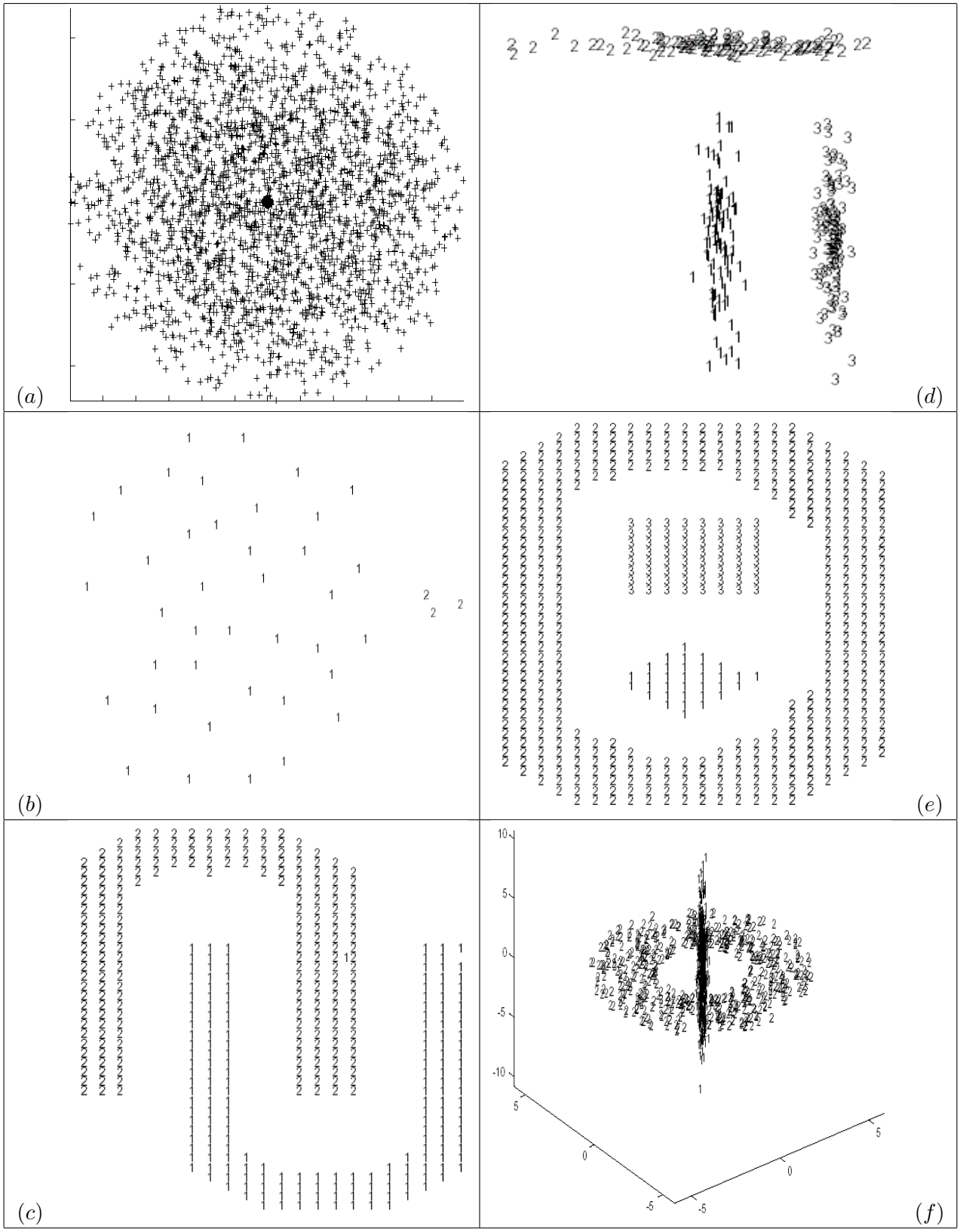


Fig. H.10: Figures extraites de [Bandyopadhyay2004], différents nuages de points bien segmentés par l'algorithme H.4.6 et de manière évidente impossible à traiter avec des méthodes apparentées aux nuées dynamiques puisque les classes obtenues ne sont pas convexes. L'image (a) permet de vérifier qu'un nuage compact distribué selon une loi normale n'est pas segmenté. L'image (b) représente un nuage composée de deux classes bien segmentées. Les autres images montrent des problèmes où les classes ne sont plus circulaires (d) ou non convexes (c), (e), (f).

comme celle obtenue par une classification ascendante hiérarchique [7] mais cette fois-ci, l'arbre final est obtenu depuis la racine jusqu'aux feuilles. Le seuil caractérisant les cas de sur-segmentation (ici $\sqrt{X}$) est celui choisi dans l'article [Bandyopadhyay2004] permettant de traiter les cas de la figure H.10. Celui-ci peut être modifié en fonction du problème à résoudre.

Cet article précise aussi que l'algorithme peut former des classes de très petites tailles qui devront être agrégées avec leurs voisines à moins que celles-ci ne soient trop éloignées, la distance entre classes étant ici la distance minimum entre leurs éléments. La règle choisie dans l'article [Bandyopadhyay2004] est que une classe sera unie à sa voisine si le diamètre de la première est inférieur à μ fois la distance qui les sépare, avec $\mu = 3 \geqslant 2$. Ce paramètre peut différer selon les problèmes.

H.5 Prolongations

H.5.1 Classe sous-représentée

Ce paragraphe regroupe quelques pistes de lecture. Les remarques qui suivent s'appliquent de préférence à une classification supervisée mais peuvent être étendues au cas non supervisé. Le premier article [Barandela2003] résume les idées concernant le cas d'un problème de classification incluant une classe sous-représentée. Par exemple, pour un problème à deux classes A et B lorsque A regroupe 98% des exemples, répondre A quelque soit l'exemple correspond à une erreur de 2%. Avec plus de 2% d'erreur, une méthode de classification serait moins performante et pourtant les classes sous-représentées favorise cette configuration. Diverses méthodes sont utilisées pour contrecarrer cet inconvénient comme la pondération des exemples sous-représentés, la multiplication de ces mêmes exemples, bruitées ou non bruitées ou encore la réduction des classes sur-représentées à un échantillon représentatif. Cette dernière option est celle discutée par l'article [Barandela2003] qui envisage différentes méthodes de sélection de cet échantillon.

H.5.2 Apprentissage d'une distance

Jusqu'à présent, seule la classification a été traitée mais on peut se demander quelle est la distance la mieux adaptée à une classification. La distance euclidienne accorde un poids égal à toutes les dimensions d'un vecteur. On peut se demander quelle est la pondération optimale pour un problème de classification donné. On définit une distance d_W avec $W = (W_1, \ldots, W_d)$ pondérant les dimensions de manière non uniforme :

$$d_W\left(X^1, X^2\right) = \sum_{k=1}^{d} W_k^2 \left(X_k^1 - X_k^2\right)^2 \tag{H.34}$$

Il reste à déterminer le vecteurs de poids $W = (W_1, \ldots, W_d)$ en s'inspirant par exemple de la méthode développée par [Waard1995]. On considère P vecteurs aussi appelés prototypes et notés $\left(X^1, \ldots, X^p\right)$ extrait du nuage $\left(X^1, \ldots, X^N\right)$. On note ensuite pour tout $p \in \{1, \ldots, P\}$:

$$y_p\left(X\right) = \frac{1}{1 + \exp\left(d_W\left(X, X^p\right) + b\right)} \tag{H.35}$$

On cherche à minimiser le critère :

7. Annexes : voir paragraphe H.4.4, page 363

$$E = \sum_{(p,l) \in A} \left(y_p \left(X_l \right) - d_{pl} \right)^2 \text{ où } A = \{1, \ldots, P\} \times \{1, \ldots, N\} \tag{H.36}$$

Cette minimisation peut être effectuée par une descente de gradient ou dans un algorithme similaire à ceux utilisés pour l'apprentissage des réseaux de neurones (voir paragraphe C.4). Chaque prototype X_p appartient à une classe C_p, les coefficients $d_{pl} \in [0,1]$ sont choisis de manière à décrire l'appartenance du vecteur X_l à la classe C_p.

Cette classification pourrait être obtenue à partir d'une classification non supervisée (centres mobiles, classification ascendante hiérarchique) mais cela suppose de disposer déjà d'une distance (comme celle par exemple décrite au paragraphe 4.2.5). Il est possible de répéter le processus jusqu'à convergence, la première classification est effectuée à l'aide d'une distance euclidienne puis une seconde distance est ensuite apprise grâce à la méthode développée dans ce paragraphe. Cette seconde distance induit une nouvelle classification qui pourra à son tour définir une troisième distance. Ce processus peut être répété jusqu'à la classification n'évolue plus.

H.5.3 Classification à partir de voisinages

L'idée de cette classification est développée dans l'article [ZhangYG2004]. Elle repose sur la construction d'un voisinage pour chaque élément d'un ensemble $E = \{x_1, \ldots, x_n\}$ à classer. La classification est ensuite obtenue en regroupant ensemble les voisinages ayant une intersection commune. L'objectif étant de proposer une réponse au problème décrit par la figure H.11.

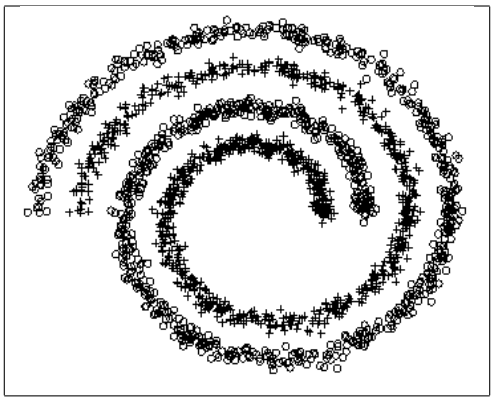


Fig. H.11: Figure extraite de [ZhangYG2004], problème classique de classification consistant à séparer deux spirales imbriquées l'une dans l'autre.

Pour chaque $x_i \in E$, on définit son voisinage local $\omega_i = \{x_{i_1}, \ldots, x_{i_K}\}$. K est le nombre de voisins et ceux-ci sont classés par ordre de proximité croissante. Par la suite, x_i sera également noté x_{i_0}. On définit ensuite la matrice de covariance S_i locale associée à ω_i :

$$m_i = \frac{1}{K+1} \sum_{k=0}^{K} x_{i_k} \text{ et } S_i = \frac{1}{K+1} \sum_{k=0}^{K} \left(x_{i_k} - m_i \right) \left(x_{i_k} - m_i \right)' \tag{H.37}$$

Le vecteur $\lambda_i = (\lambda_{i,1}, \ldots, \lambda_{i,d})'$ vérifiant $\lambda_{i,1} \geqslant \ldots \geqslant \lambda_{i,d}$, d est la dimension de l'espace vectoriel. L'*adaptibilité* a_i de l'ensemble ω_i est définie par :

$$\overline{\lambda_{i,j}} = \frac{1}{K} \sum_{t \in \{i_1,\ldots,i_K\}} \lambda_{t,j} \text{ et } a_i = \frac{1}{d} \sum_{j=1}^{d} \frac{\lambda_{i,j}}{\overline{\lambda_{i,j}}} \tag{H.38}$$

On note également :

$$E(a_i) = \frac{1}{N} \sum_{i=1}^{N} a_i \text{ et } D(a_i) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (a_i - E(a_i))^2} \tag{H.39}$$

Dans un premier temps, les voisinages déterminés par cette méthode vont être nettoyés des voisins indésirables. Ce système est souvent représenté sous forme de graphe, chaque nœud représente un élément, chaque arc détermine l'appartenance d'un élément au voisinage d'un autre. Ces graphes sont appelés "*mutual neighborhood graph*" ou *graphe des voisinages mutuels.*

Algorithme H.5.1 : nettoyage des voisinages
Les notations utilisées sont celles des expressions (H.37), (H.38), (H.39).
Etape A : estimation
Les valeurs a_i^l sont calculées pour chaque ensemble ω_i privé de x_{i_l} pour élément de l'ensemble ω_i.
Etape B : suppression
Si $a_i^l > E(a_i) + D(a_i)$, alors l'algorithme s'arrête. Sinon, l'élément x_{i_s} correspondant à la plus petite valeur x_{i_l} est supprimée de l'ensemble ω_i. On retourne ensuite à l'étape A.

La classification correspond aux composantes connexes du graphe nettoyé qui détermine par ce biais le nombre de classes. L'article suggère également d'associer à chaque élément x_i le vecteur $(x_i, \beta\lambda_i)'$ où β est un paramètre de normalisation. Le vecteur $\beta\lambda_i$ caractérise le voisinage. Ainsi, la distance entre deux points dépend à la fois de leur position et de leur voisinage. Les auteurs proposent également d'autres distances que la distance euclidienne. Il reste toutefois à déterminer les paramètres K et β.

H.5.4 Modélisation de la densité des observations

L'article [Hoti2004] présente une modélisation semi-paramétrique. Soit $Z = (X, Y)$ une variable aléatoire composée du couple (X, Y). La densité de z est exprimée comme suit :

$$f_{X,Y}(x, y) = f_{Y|X=x}(y)\, f_X(x) \tag{H.40}$$

Dans cet article, la densité $f_X(x)$ est estimée de façon non paramétrique tandis que $f_{Y|X}(y)$ est modélisée par une loi gaussienne. On note p la dimension de X et q celle de Y. On note $K_H(x) = \frac{1}{\det H} K\left(H^{-1}X\right)$ où $H \in M_p(\mathbb{R})$ est une matrice carrée définie strictement positive et K un noyau vérifiant $\int_{\mathbb{R}^p} K(x)\, dx = 1$. K peut par exemple être une fonction gaussienne. Les notations reprennent celles du paragraphe 4.3.6 (page 105). On suppose également que la variable $Y|X = x \sim \mathcal{N}(\mu(x), \sigma(x))$. Par conséquent, la densité de la variable $Z = (X, Y)$ s'exprime de la façon suivante :

$$f_{X,Y}(x,y) = \frac{f_X(x)}{\sqrt{(2\pi)^q \det \sigma^2(x)}} \exp\left(-\frac{1}{2}\left[y-\mu(x)\right] \sigma^{-1}(x) \left[y-\mu(x)\right]'\right) \tag{H.41}$$

La densité f_X est estimée avec un estimateur à noyau à l'aide de l'échantillon $(X_i, Y_i)_{1 \leqslant i \leqslant N}$:

$$\widehat{f_X}(x) = \frac{1}{N} \sum_{i=1}^{N} K_H\left(x - X_i\right) \tag{H.42}$$

On note :

$$W_H\left(x - X_i\right) = \frac{K_H\left(x - X_i\right)}{\sum\limits_{i=1}^{N} K_H\left(x - X_i\right)} \tag{H.43}$$

Les fonctions $\mu(x)$ et $\sigma(x)$ sont estimées à l'aide d'un estimateur du maximum de vraisemblance :

$$\widehat{\mu}(x) = \sum_{i=1}^{n} W_H\left(x - X_i\right) Y_i \tag{H.44}$$

$$\widehat{\sigma}(x) = \sum_{i=1}^{n} W_H\left(x - X_i\right) \left[Y_i - \widehat{\mu}(x)\right]' \left[Y_i - \widehat{\mu}(x)\right] \tag{H.45}$$

Le paragraphe 4.3.6 (page 105) discute du choix d'une matrice H appropriée (voir également [Silverman1986]). En ce qui concerne le problème de classification étudiée ici, la variable X est simplement discrète et désigne la classe de la variable Y. Cette méthode est proche de celle développée au paragraphe H.4.1 à la seule différence que l'information X_i est ici connue. L'intérêt de cette méthode est sa généralisation au cas où X est une variable continue comme par exemple un vecteur formé des distances du point X_i aux centres des classes déterminées par un algorithme de classification non supervisée. L'article [Hoti2004] discute également d'un choix d'une densité f_X paramétrique.

Annexe I

Classification supervisée

Cette annexe recense différents moyens d'effectuer une classification supervisée. Cette tâche consiste à étiqueter un élément x sachant qu'on connaît déjà cet étiquetage pour un certain nombre d'éléments $(x_1, \ldots, x_N)$ dont les labels sont $(c(x_1), \ldots, c(x_N))$.

I.1 Plus proches voisins

Cette méthode est la plus simple puisqu'elle consiste à associer à x, l'élément à classer, le label $c(x_{i^*})$ de l'élément le plus proche x_{i^*} dans l'ensemble $(x_1, \ldots, x_N)$. Ceci mène à l'algorithme de classification suivant :

> **Algorithme I.1.1 : 1-PPV ou plus proche voisin**
> Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit x un élément à classer, on cherche à déterminer la classe $\hat{c}(x)$ associée à x. On définit x_{i^*} comme étant :
>
> $$x_{i^*} = \underset{i \in \{1, \cdots, N\}}{\arg\min}\ d(x_i, x)$$
>
> Alors : $\hat{c}(x) = c(x_i^*)$

Cet algorithme est souvent appelé *1-PPV* (ou *1-NN* pour Nearest Neighbors). Il existe une version amé-

liorée *k-PPV* qui consiste à attribuer à x la classe la plus représentée parmi ses k plus proches voisins.

Algorithme I.1.2 : k-PPV ou k plus proches voisins
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit x un élément à classer, on cherche à déterminer la classe $c(x)$ associée à x. On définit l'ensemble S_k^* incluant les k-plus proches voisins de x, cet ensemble vérifie :

$$card(S_k^*) = 0 \text{ et } \max_{y \in S_k^*} d(y,x) \leqslant \min_{y \in X - S_k^*} d(y,x)$$

On calcule les occurrences $f(i)$ de chaque classe i dans l'ensemble S_k^* :

$$f(i) = \sum_{y \in S_k^*} \omega(x,y) \ \mathbb{1}_{\{c(y)=i\}} \tag{I.1}$$

On assigne alors à x la classe $c(x)$ choisie dans l'ensemble :

$$\hat{c}(x) \in \arg\max_{i \in \mathbb{N}} f(i)$$

Dans sa version la plus simple, la fonction $\omega(x,y)$ utilisée lors du calcul de la contribution f (I.1) est constante. Mais il est possible de lui affecter une valeur tenant compte de la proximité entre x et y. La table I.1 donne quelques exemples de contributions possibles.

fonction constante	$\omega(x,y) = 1$
distance inverse	$\omega(x,y) = \frac{1}{1+d(x,y)}$
noyau	$\omega(x,y) = \exp\left(-d^2(x,y)\right)$

Tab. I.1: Exemple de contribution $w(x,y)$ pour l'algorithme I.1.2 des k-PPV. Ces fonctions sont toutes décroissantes (strictement ou non) par rapport à la distance d.

L'inconvénient majeur de la méthode des plus proches voisins est sa longueur puisqu'elle implique le calcul des distances entre x et chacun des éléments de l'ensemble $(x_1, \ldots, x_N)$. C'est pourquoi de nombreuses méthodes d'optimisation ont été développées afin d'accélérer ce processus. Il est possible d'optimiser le calcul de la distance ou bien d'éviter un trop nombre de calculs en utilisant des éléments pivots [1]. L'optimisation de la vitesse est souvent préconisée lorsque l'espace métrique E n'est pas vectoriel, comme un espace de suites finies. En revanche, l'utilisation de pivots de manière à éviter l'exploration de la totalité de l'ensemble X est valable pour tout espace métrique. Ces méthodes sont l'objet de l'annexe K.

I.2 Support Vector Machines (SVM)

L'algorithme I.1.2 utilise une contribution notée ω lors du calcul de f (I.1). Si celle-ci est définie de manière explicite, on reste dans le cadre des plus proches voisins. En revanche, si celle-ci est estimée à partir d'un échantillon supposé représentatif du problème de classification à résoudre, on se place dans le cadre des *Support Vector Machines*. Ce formalisme introduit par Vapnik ([Vapnik1998]) n'est pas simplement un prolongement de la méthode des plus proches voisins mais peut aussi être interprêté comme la recherche

1. Annexes : voir paragraphe K, page 390

du meilleur hyperplan de séparation entre deux classes. Cette méthode est présentée plus en détail par l'annexe D [2].

I.3 Réseaux de neurones

Cette méthode est présentée plus en détail aux paragraphes C.1.5 (page 196) et C.5 (page 217) [3]. Elle permet de construire une fonction $f(x) = y = (y_1, \ldots, y_C) \in \mathbb{R}^C$ où C est le nombre de classes, $y_i \in [0, 1]$ et $\sum_1^C y_i = 1$. Chaque sortie y_i du réseau de neurones correspond à la probabilité que le vecteur x appartient à la classe i. Contrairement aux deux méthodes précédentes, les réseaux de neurones permettent de construire une fonction f indépendante du nombre d'exemples permettant de l'estimer. Néanmoins, les paramètres de cette fonction ne sont plus aussi interprétables que les contributions évoquées aux paragraphes I.1 et I.2. Ceci explique que le fort intérêt de ces modèles depuis les années 1980 au milieu des années 1990 ait décru au profit d'autres solutions comme les SVM.

I.4 Learning Vector Quantization (LVQ)

Cette méthode est souvent associé à des méthodes de classification par plus proches voisins évoquées dans l'annexe K. Lors de la classification d'un élément, on recherche dans un ensemble le plus proche élément et on attribue à l'élément à classer la classe de l'élément trouvé. Alors que l'annexe K cherche à accélérer la recherche de l'élément le plus proche, la méthode LVQ essaye de résumer l'information au travers de prototypes. Plus simplement, les méthodes abordées ici permettent tente réduire au minimum l'ensemble dans lequel seront cherchés les voisins sans changer ou sans trop changer le résultat de la classification.

En ce qui concerne les nuées dynamiques, les prototypes sont les centres des classes déterminées par l'algorithme des centres mobiles. Pour les différentes versions LVQ qui suivent, les prototypes doivent représenter au mieux une classification imposée. L'article [Bezdek2001] propose une revue récente de ces méthodes que reprend en partie seulement un article [Kim2003] sur lequel s'appuie les paragraphes qui suivent.

I.4.1 Principe

Lors de l'algorithme I.1.1 qui permet de classer un élément x en l'associant à la même classe que son plus proche voisin, il faut calculer toutes les distances de x aux voisins possibles X. Les méthodes LVQ ont pour objectif de réduire l'ensemble X à une taille raisonnable en utilisant l'information de la classe. Après réduction, l'algorithme de classification doit retourner les mêmes réponses. Par conséquent, l'algorithme

2. Annexes : voir paragraphe D, page 238
3. Annexes : voir paragraphe C, page 190

suivant I.4.1 doit retourner les mêmes réponses que la méthode 1-ppv I.1.1.

Algorithme I.4.1 : 1-PPV avec LVQ
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit x un élément à classer, on cherche à déterminer la classe $c(x)$ associée à x.

Etape A : LVQ
On réduit l'ensemble X à un ensemble de prototypes X' qui n'est pas forcément inclus dans X. On note $X' = (x'_1, \ldots, x'_n)$ avec de préférence $n << N$. La classe de l'élément x'_i est toujours notée $c(x'_i)$. Les algorithmes effectuant cette réduction sont présentées dans les paragraphes qui suivent comme I.4.2 ou I.4.3.

Etape B : classification
On définit x'_{i^*} comme étant :

$$x'_{i^*} = \underset{i \in \{1, \cdots, n\}}{\arg\min} \; d\left(x'_i, x\right)$$

Alors : $\hat{c}(x) = c(x'_{i^*})$

Les paragraphes qui suivent présentent des algorithmes permettant de calculer un ensemble X' satisfaisant à l'étape A et le plus réduit possible. Cette étape A est un fait un prétraitement, elle n'est effectuée qu'une seule fois tandis que l'étape B intervient pour chaque nouvel élément à classer. L'ensemble X' est appelé l'ensemble des *prototypes*. Les chapitres qui suivent concernent essentiellement les espaces vectoriels excepté pour le paragraphe I.4.2. Pour des espaces métriques non vectoriels, l'annexe K présente d'autres méthodes de sélection de prototypes [4].

I.4.2 Condensed nearest neighbors rule (CNN)

Cette méthode est développée dans [Hart1968]. Elle consiste à construire un ensemble X' à partir des éléments de X. Un premier élément est choisi aléatoirement puis placé dans X'. On parcourt ensuite l'ensemble X, pour chaque élément x, on applique l'algorithme I.4.1. Si le résultat ne correspond pas à la

4. Annexes : voir paragraphe K.3.3, page 412

classe $c(x)$, cet élément est ajouté à l'ensemble X'. Ceci mène à l'algorithme suivant :

Algorithme I.4.2 : CNN
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E.
Etape A : initialisation
Soit x un élément de X, $X' \longleftarrow \{x\}$ et $Y \longleftarrow \{x\}$.

Etape B : construction de X'
tant que $(Y \neq X)$ **faire**
Soit $x \in X - Y$, on applique l'étape B de l'algorithme I.4.1 à l'élément x.
si $\hat{c}(x) \neq c(x)$ **alors**
$X' \longleftarrow X' \cup \{x\}$
fin si
$Y \longleftarrow Y \cup \{x\}$
fin tant que

Cet algorithme n'impose pas un nombre précis de prototypes. De plus, puisque $X' \subset X$, cette méthode est applicable à tout espace métrique, il ne nécessite pas qu'il soit vectoriel. Toutefois, l'algorithme est sensible à l'ordre dans lequel sont traités les éléments de X.

I.4.3 Prototype for nearest neighbors (PNN)

Cette méthode est développée dans [Chang1974]. Contrairement à l'algorithme précédent I.4.2, l'ensemble X' n'est plus inclus dans X et est construit de manière à obtenir autant que faire ce peu des barycentres des classes. Il ne s'applique donc qu'à des espaces vectoriels. Au départ, tous les éléments de X sont considérés comme des prototypes. Puis les plus proches d'entre eux appartenant à la même classe vont

être agrégés si aucune erreur de classification n'est constatée.

Algorithme I.4.3 : PNN
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E.
Etape A : initialisation
On définit les ensembles $A \longleftarrow \emptyset$ et $B \longleftarrow X$ ainsi que la suite $(p(x_1), \ldots, p(x_N))$ telle que $\forall i,\ p(x_i) = 1$. $t \longleftarrow 0$ et $\epsilon_0 \longleftarrow \infty$.
Etape B : construction de B
tant que $(B \neq \emptyset)$ **faire**
$m \longleftarrow 0$. On définit $x_A \in A$ et $x_B \in B$ tels que :

$$d(x_A, x_B) = \min \{d(x, y) \mid x \in A, y \in B\}$$

si $c(x_A) \neq c(x_B)$ **alors**
$B \longleftarrow B - \{x_B\}$ et $A \longleftarrow A \cup \{x_B\}$
sinon
$x \longleftarrow \dfrac{p(x_A)\ x_A + p(x_B)\ x_B}{p(x_A) + p(x_B)}$
$p(x) \longleftarrow p(x_A) + p(x_B)$
On note ϵ_t le taux de classification obtenu avec l'ensemble de prototypes $X' = A \cup \{x\}$.
si $\epsilon_t > \epsilon_{t-1}$ **alors**
$B \longleftarrow B - \{x_B\}$ et $A \longleftarrow A \cup \{x_B\}$
sinon
$B \longleftarrow B - \{x_B\}$ et $A \longleftarrow [A - \{x_A\}] \cup \{x\}$
$m \longleftarrow m + 1$
fin si
fin si
fin tant que
Etape C : terminaison
si $m > 0$ **alors**
$B \longleftarrow A$ et $A \longleftarrow \emptyset$.
On retourne à l'étape B.
sinon
L'algorithme s'arrête et l'ensemble cherché $X' \longleftarrow A$.
fin si

L'article [Bezdek2001] suggère de ne considérer lors de l'étape B que des paires (x_A, x_B) appartenant à une même classe de manière à ce que le résultat obtenu soit plus consistent. Ce second algorithme est plus lent que l'algorithme I.4.2 mais la remarque à propos l'ordre d'insertion ne le concerne plus.

I.4.4 LVQ1, ..., LVQ4

Les LVQ ont été introduits dans [Linde1980], adaptés par la suite par Kohonen à la reconnaissance des formes ([Kohonen1982], [Kohonen1995]). Historiquement, le premier algorithme LVQ1 est dû à Kohonen et permet de déterminer un nombre fixé de prototypes, contrairement aux deux algorithmes I.4.2 et I.4.3

des paragraphes précédents ne nécessitant aucun a priori sur leur nombre.

Algorithme I.4.4 : LVQ1
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit $P = (p_1, \ldots, p_k)$ k prototypes tirés aléatoirement. On associe une classe $\overline{c}(p_k)$ à chaque prototype. La suite (α_t) est une suite positive vérifiant : $\sum_t \alpha_t = \infty$ et $\sum_t \alpha_t^2 < \infty$. Enfin, $t \longleftarrow 0$.
Etape A : meilleur prototype
$t \longleftarrow t+1$
On choisit aléatoirement un élément $x \in X$. On détermine $p^* = \arg\min_{p \in P} d(x,p)$.
Etape B : mise à jour
$$p^* \longleftarrow p^* + \begin{cases} \alpha_t (x - p^*) & \text{si } \overline{c}(p^*) = c(x) \\ -\alpha_t (x - p^*) & \text{si } \overline{c}(p^*) \neq c(x) \end{cases}$$
On retourne à l'étape A tant que les prototypes continuent d'évoluer.

Le nombre de prototypes est fixé au départ ainsi que la classe qui est associée à chacun d'eux. Cet algorithme est souvent utilisé avec autant de prototypes qu'il y a de classes. La suite (α_t) est en principe une suite décroissante mais qui peut être choisie de telle manière que :

$$\alpha_{t+1} = \begin{cases} \frac{\alpha_t}{1+\alpha_t} & \text{si } \overline{c}(p^*) = c(x) \\ \frac{\alpha_t}{1-\alpha_t} & \text{si } \overline{c}(p^*) \neq c(x) \end{cases} \tag{I.2}$$

Le pas d'apprentissage α_t croît si le prototype le plus proche est d'une classe différente de celle de l'élément x. Cette version de l'algorithme LVQ1 est appelé *Optimized LVQ1*. Cette optimisation est valable pour tous les algorithmes de la famille LVQ qui suivent. La seconde version de cet algorithme propose la mise à jour simultanée de deux prototypes qui permet d'améliorer les frontières de décision.

Algorithme I.4.5 : LVQ2
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit $P = (p_1, \ldots, p_k)$ k prototypes tirés aléatoirement. On associe une classe $\overline{c}(p_k)$ à chaque prototype. La suite (α_t) est une suite positive vérifiant : $\sum_t \alpha_t = \infty$ et $\sum_t \alpha_t^2 < \infty$. On pose également $t \longleftarrow 0$. Soit $w \in]0,1[$.
Etape A : meilleur prototype
$t \longleftarrow t+1$
On choisit aléatoirement un élément $x \in X$. On détermine :
$p_1^* = \arg\min \{d(x,p) \mid p \in P, \overline{p}(p) = c(x)\}$ et
$p_2^* = \arg\min \{d(x,p) \mid p \in P, \overline{p}(p) \neq c(x)\}$.
Etape B : mise à jour
si $\frac{1-w}{1+w} < \frac{d(x,p_1^*)}{d(x,p_2^*)} < \frac{1+w}{1-w}$ **alors**
$p_1^* \longleftarrow p_1^* + \alpha_t (x - p_1^*)$
$p_2^* \longleftarrow p_2^* - \alpha_t (x - p_2^*)$
fin si
On retourne à l'étape A tant que les prototypes continuent d'évoluer.

Le livre [Kohonen1995] suggère de choisir $w \in [0,2\,;\,0,3]$. L'algorithme LVQ3 qui suit propose une extension de l'algorithme LVQ2 pour des prototypes p_1^* et p_2^* appartenant à la même classe.

Algorithme I.4.6 : LVQ3
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit $P = (p_1, \ldots, p_k)$ k prototypes tirés aléatoirement. On associe une classe $\overline{c}(p_k)$ à chaque prototype. La suite (α_t) est une suite positive vérifiant : $\sum_t \alpha_t = \infty$ et $\sum_t \alpha_t^2 < \infty$. On pose également $t \longleftarrow 0$. Soit $w \in]0,1[$ et $\epsilon \in [0,1\,;\,0,5]$.
Etape A : meilleur prototype
$t \longleftarrow t+1$
On choisit aléatoirement un élément $x \in X$. On détermine :
$p_1^* = \arg\min \{d(x,p) \mid p \in P, \overline{p}(p) = c(x)\}$ et
$p_2^* = \arg\min \{d(x,p) \mid p \in P, p_2^* \neq p_1^*\}$.
Etape B : mise à jour
si $\overline{c}(p_1^*) \neq \overline{c}(p_2^*)$ **alors**
si $\frac{1-w}{1+w} < \frac{d(x,p_1^*)}{d(x,p_2^*)} < \frac{1+w}{1-w}$ **alors**
$p_1^* \longleftarrow p_1^* + \alpha_t (x - p_1^*)$
$p_2^* \longleftarrow p_2^* - \alpha_t (x - p_2^*)$
fin si
sinon
$p_1^* \longleftarrow p_1^* + \epsilon\alpha_t (x - p_1^*)$
$p_2^* \longleftarrow p_2^* + \epsilon\alpha_t (x - p_2^*)$
fin si On retourne à l'étape A tant que les prototypes continuent d'évoluer.

L'algorithme LVQ4 est la version la plus récente. Il s'inspire de l'algorithme LVQ1 mais modifie le poids de l'apprentissage α_t de manière plus pertinente.

Algorithme I.4.7 : LVQ4
Soit $X = (x_1, \ldots, x_N) \subset E$ un ensemble d'éléments d'un espace métrique quelconque, soit $(c(x_1), \ldots, c(x_N))$ les classes associées à chacun des éléments de X. On note d la distance définie sur l'espace métrique E. Soit $P_t = (p_{t,1}, \ldots, p_{t,k})$ k prototypes tirés aléatoirement. On associe une classe $\overline{c}(p_{t,k})$ à chaque prototype. La suite (α_t) est une suite positive vérifiant : $\sum_t \alpha_t = \infty$ et $\sum_t \alpha_t^2 < \infty$. On pose également $t \longleftarrow 0$. On suppose que $\forall t,\ \alpha_t \in]0,1[$. Soit $\lambda > 1$.
Etape A : meilleur prototype
$t \longleftarrow t+1$
On choisit aléatoirement un élément $x \in X$. On détermine $p_t^* = \arg\min_{p \in P_t} d(x,p)$.
Etape B : mise à jour
$p_t^* \longleftarrow p_t^* + s(p_t^*)\ \alpha_t (x - p_t^*)$ où

$$s(p_t^*) = \begin{cases} \lambda & \text{si } \overline{c}(p_t^*) = c(x) \text{ et } M(p_t^*) = 0 \\ \frac{B(p_t^*)}{M(p_t^*)} & \text{si } \overline{c}(p_t^*) = c(x) \text{ et } M(p_t^*) > 0 \\ -1 & \text{si } \overline{c}(p_t^*) \neq c(x) \end{cases}$$

$B(p_t^*)$ représente le nombre d'exemples bien classés avec le prototype p_t^* tandis que $M(p_t^*)$ est le nombre d'exemples mal classés. On retourne à l'étape A tant que les prototypes continuent d'évoluer.

L'inconvénient de cet algorithme est le calcul coûteux de $B\left(p_t^*\right)$ et $M\left(p_t^*\right)$. L'évaluation des nombres $B\left(p\right)$ et $M\left(t\right)$ pour $p \in P_t$ devrait être effectuée à chaque itération t, c'est-à-dire à chaque fois qu'un prototype est actualisé. Afin d'accélérer l'algorithme, cette évaluation n'est pas effectuée à chaque itération mais toutes les T itérations où T serait la période de mise à jour. Durant une période, ces nombres peuvent être considérés comme constants où évoluer en tenant de compte leur passé. Différentes variantes de l'algorithme $LVQ4$ sont proposées et discutées dans l'article [Vakil2003].

I.5 Prolongations

I.5.1 Liens entre LVQ et la rétropropagation

L'article [Frasconi1997] met en rapport l'algorithme LVQ1 (I.4.4) et l'algorithme de rétropropagation C.5.4 dans un réseau de neurones sont les fonctions de transfert sont des fonctions à base radiale ou RBF [5]. Ce réseau de neurones contient autant de neurones sur la couche cachée qu'il y a de prototypes dans l'algorithme LVQ1. La sortie des neurones cachés est donnée par :

$$z_i = \exp\left[-\frac{\|p-x\|^2}{\sigma^2}\right]$$

p est un prototype, x est un élément, l'élément pour lequel on évalue les sorties du réseau de neurones. L'article [Frasconi1997] que lorsque $\sigma \longrightarrow 0$, on construit ensuite le nombre :

$$z'_i = \frac{z_i}{\sum\limits_i z_i}$$

Lorsque $\sigma \longrightarrow 0$, le vecteur $(z'_1, \ldots, z'_k)$ converge vers un vecteur presque nul sauf pour le prototype i le plus proche. De même, lorsque $\sigma \longrightarrow 0$, une itération d'un apprentissage par rétropropagation d'un tel réseau de neurones est équivalente à une itération de l'algorithme LVQ1.

5. Annexes : voir paragraphe C.6.2, page 224

Annexe J

Distance d'édition

Les distances d'édition permettent de comparer deux mots entre eux ou plus généralement deux séquences de symboles entre elles. L'usage le plus simple est de trouver, pour un mot mal orthographié, le mot le plus proche dans un dictionnaire, c'est une option proposée dans la plupart des traitements de texte. La distance présentée est la distance de Levenstein (voir [Levenstein1966]). Elle est parfois appelée Damerau Levenstein Matching (DLM) (voir également [Damerau1964]). Cette distance fait intervenir trois opérations élémentaires :

1. comparaison entre deux caractères
2. insertion d'un caractère
3. suppression d'un caractère

Pour comparer deux mots, il faut construire une méthode associant ces trois opérations afin que le premier mot se transforme en le second mot. L'exemple J.1 utilise les mots "idstzance" et "distances", il montre une méthode permettant de passer du premier au second. La distance sera la somme des coûts associés à chacune des opérations choisies. La comparaison entre deux lettres identiques est en général de coût nul, toute autre opération étant de coût strictement positif.

mot 1	**mot 2**	**opération**	**coût**
i	d	comparaison entre "i" et "d"	1
d	i	comparaison entre "d" et "i"	1
s	s	comparaison entre "s" et "s"	0
t	t	comparaison entre "t" et "t"	0
z		suppression de "z"	1
a	a	comparaison entre "a" et "a"	0
n	n	comparaison entre "n" et "n"	0
c	c	comparaison entre "c" et "c"	0
e	e	comparaison entre "e" et "e"	0
	s	insertion de "s"	1
		somme = distance	4

Fig. J.1: Distance d'édition entre les mots "idstzance" et "distances". La succession d'opérations proposée n'est pas la seule qui permettent de construire le second mot à partir du premier mais c'est la moins coûteuse.

J.1 Définition et propriétés

J.1.1 Définition

Tout d'abord, il faut définir ce qu'est un mot ou une séquence :

Définition J.1.1 : mot
On note $\mathcal{C}$ l'espace des caractères ou des symboles. Un mot ou une séquence est une suite finie de $\mathcal{C}$. On note $\mathcal{S}_{\mathcal{C}} = \bigcup_{k=1}^{\infty} C^k$ l'espace des mots formés de caractères appartenant à $\mathcal{C}$.

On peut définir la distance d'édition :

Définition J.1.2 : distance d'édition
La distance d'édition d sur $\mathcal{S}_{\mathcal{C}}$ est définie par :

$$\begin{array}{rccl} d: & \mathcal{S}_{\mathcal{C}} \times \mathcal{S}_{\mathcal{C}} & \longrightarrow & \mathbb{R}^+ \\ & (m_1, m_2) & \longrightarrow & \min\limits_{\substack{O \text{ séquence} \\ \text{d'opérations}}} d(m_1, m_2, O) \end{array}$$

La distance J.1.2 est le coût de la transformation du mot m_1 en m_2 la moins coûteuse. Il reste à démontrer que cette distance en est bien une (paragraphe J.1.2) puis à proposer une méthode de calcul plus rapide que celle suggérée par cette définition.

J.1.2 Propriétés

Ce paragraphe a pour objectif de démontrer que la distance définie en J.1.2 en est bien une.

Définition J.1.3 : distance entre caractères
Soit $\mathcal{C}' = \mathcal{C} \bigcup \{.\}$ l'ensemble des caractères ajouté au caractère vide ".".
On note $c : (\mathcal{C}')^2 \longrightarrow \mathbb{R}^+$ la fonction coût définie comme suit :

$$\forall (x,y) \in \left(\mathcal{C}'\right)^2,\ c(x,y) \text{ est le coût } \begin{cases} \text{d'une comparaison} & \text{si } (x,y) \in (\mathcal{C})^2 \\ \text{d'une insertion} & \text{si } (x,y) \in (\mathcal{C}) \times \{.\} \\ \text{d'une suppression} & \text{si } (x,y) \in \{.\} \times (\mathcal{C}) \\ 0 & \text{si } (x,y) = (\{.\}, \{.\}) \end{cases} \quad \text{(J.1)}$$

On note $\mathcal{S}^2_{\mathcal{C}'} = \bigcup_{n=1}^{\infty} \left(\mathcal{C}'^2\right)^n$ l'ensemble des suites finies de $\mathcal{C}'$.

Pour modéliser les transformations d'un mot vers un autre, on définit pour un mot m un *mot acceptable* :

Définition J.1.4 : mot acceptable
Soit $m = (m_1, \ldots, m_n)$ un mot tel qu'il est défini en J.1.1. Soit $M = (M_i)_{i \geqslant 1}$ une suite infinie de caractères, on dit que M est un mot acceptable pour m si et seulement si la sous-suite extraite de M contenant tous les caractères différents de $\{.\}$ est égal au mot m. On note $acc(m)$ l'ensemble des mots acceptables pour le mot m.

Par conséquent, tout mot acceptable m' pour le mot m est égal à m si on supprime les caractères $\{.\}$ du mot m'. En particulier, à partir d'un certain indice, m' est une suite infinie de caractères $\{.\}$. Il reste alors à exprimer la définition J.1.2 en utilisant les mots acceptables :

Définition J.1.5 : distance d'édition
Soit c la distance définie en J.1.3, la distance d'édition d sur $\mathcal{S}_\mathcal{C}$ est définie par :

$$\begin{array}{rccl} d: & \mathcal{S}_\mathcal{C} \times \mathcal{S}_\mathcal{C} & \longrightarrow & \mathbb{R}^+ \\ & (m_1, m_2) & \longrightarrow & \min \left\{ \sum_{i=1}^{+\infty} c\left(M_1^i, M_2^i\right) \mid (M_1, M_2) \in acc\,(m_1) \times acc\,(m_2) \right\} \end{array} \tag{J.2}$$

Il est évident que la série $\sum_{i=1}^{+\infty} c\left(M_1^i, M_2^i\right)$ est convergente. La distance de caractères définie en J.1.3 implique que les distance d'édition définies en J.1.2 et J.1.5 sont identiques.

Théorème J.1.6 : distance d'édition
Soit c et d les fonctions définies respectivement par (J.1) et (J.2), alors :

$$c \text{ est une distance sur } \mathcal{C}' \Longleftrightarrow d \text{ est une distance sur } \mathcal{S}_\mathcal{C}$$

Démonstration (théorème J.1.6) :

Partie A (démonstration de J.1.6)

On cherche d'abord à démontrer que :

$$c \text{ est une distance sur } \mathcal{C}' \Longleftarrow d \text{ est une distance sur } \mathcal{S}_\mathcal{C}$$

Cette assertion est évidente car, si (m_1, m_2) sont deux mots de un caractère, la distance d sur $\mathcal{S}_\mathcal{C}$ définit alors la distance c sur $\mathcal{C}'$.

Partie B (démonstration de J.1.6)

On cherche démontrer que :

$$c \text{ est une distance sur } \mathcal{C}' \Longrightarrow d \text{ est une distance sur } \mathcal{S}_\mathcal{C}$$

Soient deux mots (m_1, m_2), soit $(M_1, M_2) \in acc\,(m_1) \times acc\,(m_2)$, comme c est une distance sur $\mathcal{C}'$:

$$d\,(M_1, M_2) = d\,(M_2, M_1)$$

D'où, d'après la définition J.1.5 :

$$d\,(m_1, m_2) = d\,(m_2, m_1) \tag{J.3}$$

Soit $(N_1, N_2) \in acc\,(m_1) \times acc\,(m_2)$ tels que :

$$d\,(m_1, m_2) = d\,(N_2, N_1)$$

Alors :

$$\begin{aligned} d(m_1, m_2) = 0 &\Longrightarrow d(N_1, N_2) = 0 \\ &\Longrightarrow \sum_{i=1}^{+\infty} c\left(N_1^i, N_2^i\right) = 0 \\ &\Longrightarrow \forall i \geqslant 1,\ N_1^i = N_2^i \\ &\Longrightarrow N_1 = N_2 \\ d(m_1, m_2) = 0 &\Longrightarrow m_1 = m_2 \end{aligned} \tag{J.4}$$

Il reste à démontrer l'inégalité triangulaire. Soient trois mots (m_1, m_2, m_3), on veut démontrer que :

$$d(m_1, m_3) \leqslant d(m_1, m_2) + d(m_2, m_3)$$

On définit :

$$\begin{aligned} (N_1, N_2) \in acc(m_1) \times acc(m_2) &\text{ tels que } d(m_1, m_2) = d(N_1, N_2) \\ (P_2, P_3) \in acc(m_2) \times acc(m_3) &\text{ tels que } d(m_2, m_3) = d(P_2, P_3) \\ (O_1, O_3) \in acc(m_1) \times acc(m_3) &\text{ tels que } d(m_1, m_3) = d(O_1, O_3) \end{aligned}$$

Mais il est possible, d'après la définition J.1.4 d'insérer des caractères $\{.\}$ dans les mots $N_1, N_2, P_2, P_3, O_1, O_3$ de telle sorte qu'il existe $(M_1, M_2, M_3) \in acc(m_1) \times \in acc(m_2) \times \in acc(m_3)$ tels que : (voir figure J.2)

$$\begin{aligned} d(m_1, m_2) &= d(M_1, M_2) \\ d(m_2, m_3) &= d(M_2, M_3) \\ d(m_1, m_3) &= d(M_1, M_3) \end{aligned}$$

Or comme la fonction c est une distance sur $\mathcal{C}'$, on peut affirmer que :

$$d(M_1, M_3) \leqslant d(M_1, M_2) + d(M_2, M_3)$$

D'où :

$$d(m_1, m_3) \leqslant d(m_1, m_2) + d(m_2, m_3) \tag{J.5}$$

Les assertions (J.3), (J.4), (J.5) montrent que d est bien une distance.

M_1	i	d		t	z	a	n	c	e	
M_2				t		o	n	c	e	
M_3	d	i	s	t		a	n	c	e	s

Fig. J.2: Démonstration du théorème J.1.6, illustration des suites M_1, M_2, M_3 pour les mots *idtzance*, *tonce*, *distances*

(J.1.6) □

Remarque J.1.7: longueur des mots

La distance d'édition J.1.5 ne tient pas compte de la longueur des mots qu'elle compare. On serait tenté de définir une nouvelle distance d'édition inspirée de la précédente :

Soit d^* la distance d'édition définie en J.1.5 pour laquelle les coûts de comparaison, d'insertion et de suppression sont tous égaux à 1.
La distance d'édition d' sur $\mathcal{S}_\mathcal{C}$ est définie par :

$$\begin{array}{rccl} d' : & \mathcal{S}_\mathcal{C} \times \mathcal{S}_\mathcal{C} & \longrightarrow & \mathbb{R}^+ \\ & (m_1, m_2) & \longrightarrow & d'(m_1, m_2) = \dfrac{d^*(m_1, m_2)}{\max\{l(m_1), l(m_2)\}} \\ & & & \text{où } l(m) \text{ est la longueur du mot } m \end{array} \tag{J.6}$$

Le tableau J.1 donne un exemple pour lequel l'inégalité triangulaire n'est pas vérifiée. La fonction d^* n'est donc pas une distance.

mot 1	mot 2	distance d^*	distance d'
$APPOLLINE$	$APPOLINE$	1	1/9
$APPOLLINE$	$APOLLINE$	1	1/9
$APOLLINE$	$APPOLINE$	2	2/8

Par conséquent :
$d(APOLLINE, APPOLINE) >$
$\quad d(APOLLINE, APPOLLINE) + d(APPOLLINE, APPOLINE)$

Tab. J.1: Distance d'édition et longueur de mots, cas particulier où la fonction d^* définie par (J.6) ne vérifie pas l'inégalité triangulaire.

J.2 Factorisation des calculs

La définition de la distance d'édition ne permet pas d'envisager le calcul de la distance dans un temps raisonnable. Il est possible néanmoins d'exprimer cette distance d'une autre manière afin de résoudre ce problème ([Wagner1974]). On définit la suite suivante :

Définition J.2.1 : distance d'édition tronquée
Soient deux mots (m_1, m_2), on définit la suite :

$$\left(d_{i,j}^{m_1,m_2}\right)_{\substack{0\leqslant i\leqslant n_1 \\ 0\leqslant j\leqslant n_2}} \left(= (d_{i,j})_{\substack{0\leqslant i\leqslant n_1 \\ 0\leqslant j\leqslant n_2}} \text{ pour ne pas alourdir les notations}\right)$$

par :

$$\left\{ \begin{array}{l} d_{0,0} = 0 \\ \\ d_{i,j} = \min \left\{ \begin{array}{lcl} d_{i-1,j-1} & + & \text{comparaison}\left(m_1^i, m_2^j\right), \\ d_{i,j-1} & + & \text{insertion}\left(m_2^j\right), \\ d_{i-1,j} & + & \text{suppression}\left(m_1^i\right) \end{array} \right. \end{array} \right.$$

Cette suite tronquée permet d'obtenir le résultat de la propriété suivante :

Propriété J.2.2 : calcul rapide de la distance d'édition
La suite définie en J.2.1 vérifie :

$$d(m_1, m_2) = d_{n_1,n_2}$$

où d est la distance d'édition définie en J.1.2 ou J.1.5.

Cette factorisation des calculs est illustrée par les tableaux de la figure J.3 (page 388).

Démonstration (propriété J.2.2) :

La démonstration s'effectue par récurrence, la définition J.2.1 est bien sûr équivalente à J.1.2 pour des mots de longueur un. On suppose donc que ce résultat est vrai pour un couple de mots (m_1, m_2) de longueur (l_1, l_2) vérifiant $l_1 \leqslant i$ et $l_2 \leqslant j$ avec au plus une égalité. Soit m un mot, on note n le nombre de lettres qu'il contient. On note $m(l)$ le mot formé des l premières lettres de m. Alors :

$$\begin{aligned} d_{i,j}^{m_1,m_2} &= d(m_1(i), m_2(j)) \\ d(m_1(i), m_2(j)) &= \min \left\{ \begin{array}{lcl} d(m_1(i-1), m_2(j-1)) & + & \text{comparaison}(m_{1,i}, m_{2,j}), \\ d(m_1(i), m_2(j-1)) & + & \text{insertion}(m_{2,j}), \\ d(m_1(i-1), m_2(j)) & + & \text{suppression}(m_{1,i}) \end{array} \right\} \end{aligned}$$

(J.2.2) □

Le calcul factorisé de la distance d'édition entre deux mots de longueur l_1 et l_2 a un coût de l'ordre $O(l_1 l_2)$. Il est souvent illustré par un tableau comme celui de la figure J.3 qui permet également de retrouver la meilleure séquence d'opérations permettant de passer du premier mot au second.

J.3 Extension de la distance d'édition

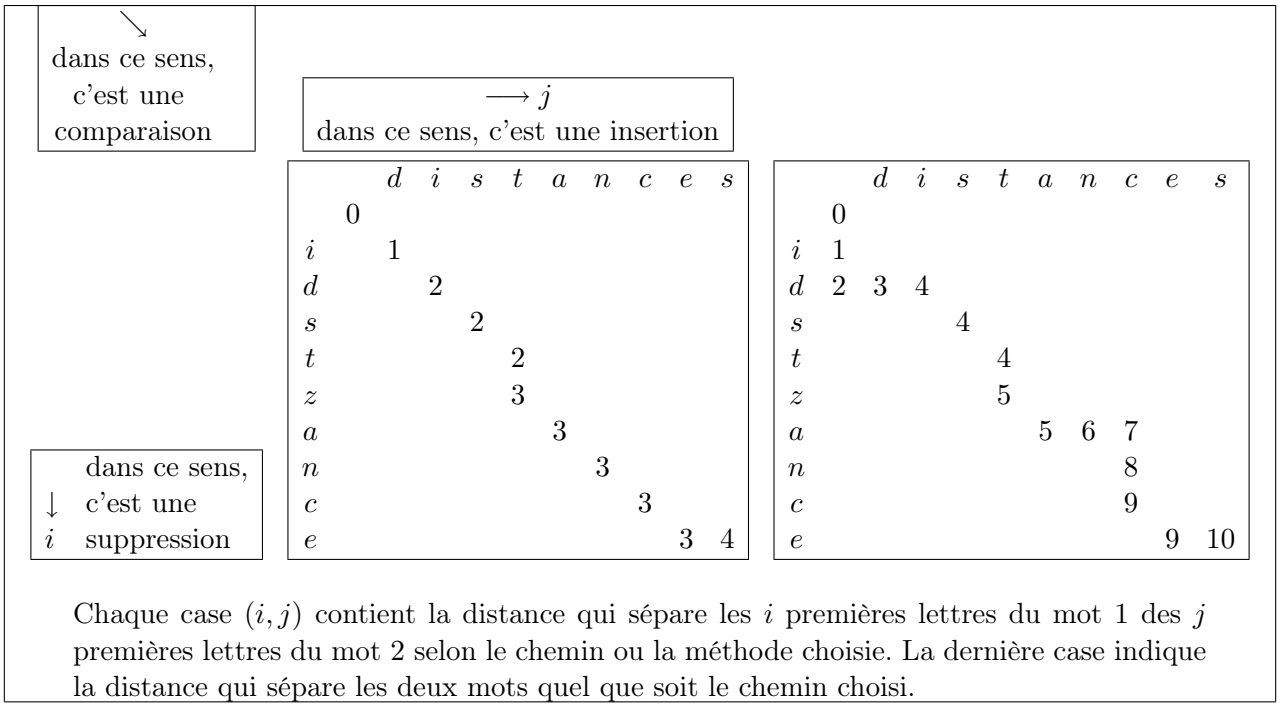
Jusqu'à présent, seuls trois types d'opérations ont été envisagés pour constuire la distance d'édition, tous trois portent sur des caractères et aucunement sur des paires de caractères. L'article [Kripasundar1996] (voir aussi [Seni1996]) suggère d'étendre la définition J.2.1 aux permutations de lettres :

Définition J.3.1 : distance d'édition tronquée étendue
Soit deux mots (m_1, m_2), on définit la suite :

$$\left(d_{i,j}^{m_1,m_2}\right)_{\substack{0 \leqslant i \leqslant n_1 \\ 0 \leqslant j \leqslant n_2}} \left(= (d_{i,j})_{\substack{0 \leqslant i \leqslant n_1 \\ 0 \leqslant j \leqslant n_2}} \text{ pour ne pas alourdir les notations} \right)$$

par :

$$\left\{ \begin{array}{l} d_{0,0} = 0 \\ d_{i,j} = \min \left\{ \begin{array}{lcl} d_{i-1,j-1} & + & \text{comparaison}\left(m_1^i, m_2^j\right), \\ d_{i,j-1} & + & \text{insertion}\left(m_2^j, i\right), \\ d_{i-1,j} & + & \text{suppression}\left(m_1^i, j\right), \\ d_{i-2,j-2} & + & \text{permutation}\left(\left(m_1^{i-1}, m_1^i\right), \left(m_2^{j-1}, m_2^j\right)\right) \end{array} \right\} \end{array} \right.$$



		d	i	s	t	a	n	c	e	s
	0									
i		1								
d			2							
s				2						
t					2					
z					3					
a						3				
n							3			
c								3		
e									3	4

		d	i	s	t	a	n	c	e	s
	0									
i	1									
d	2	3	4							
s				4						
t					4					
z					5					
a						5	6	7		
n								8		
c								9		
e									9	10

Fig. J.3: Chemins possibles afin de comparer les mots "idstzance" et "distances" avec une distance d'édition

La distance d'édition cherchée est toujours $d(m_1, m_2) = d_{n_1,n_2}$ mais la démonstration du fait que d est bien une distance ne peut pas être copiée sur celle du théorème J.1.6 mais sur les travaux présentés dans l'article [Wagner1974].

J.4 Apprentissage d'une distance d'édition

L'article [Waard1995] suggère l'apprentissage des coûts des opérations élémentaires associées à une distance d'édition (comparaison, insertion, suppression, permutation, ...). On note l'ensemble de ces coûts ou paramètres $\Theta = (\theta_1, \ldots, \theta_n)$. On considère deux mots X et Y, la distance d'édition $d(X, Y)$ est une fonction linéaire des coûts. Soit $D = ((X_1, Y_1), \ldots, (X_N, Y_N))$ une liste de couple de mots pour lesquels le résultat de la distance d'édition est connu et noté $(c_1, \ldots, c_N)$, il est alors possible de calculer une erreur s'exprimant sous la forme :

$$E = \sum_{i=1}^{N} \left(d(X_i, Y_i) - c_i \right)^2 = \sum_{i=1}^{N} \left(\sum_{k=1}^{n} \alpha_{ik}(\Theta)\ \theta_k - c_i \right)^2 \tag{J.7}$$

$$\tag{J.8}$$

Les coefficients $\alpha_{ik}(\Theta)$ dépendent des paramètres Θ car la distance d'édition correspond au coût de la transformation de moindre coût d'après la définition J.1.5, $\alpha_{ik}(\Theta)$ correspond au nombre de fois que le paramètre θ_k intervient dans la transformation de moindre coût entre X_i et Y_i. Cette expression doit être minimale afin d'optenir les coûts Θ optimaux. Toutefois, les coûts θ_k sont tous strictement positifs et plutôt que d'effectuer une optimisation sous contrainte, ces coûts sont modélisés de la façon suivante :

$$E = \sum_{i=1}^{N} \left(\sum_{k=1}^{n} \alpha_{ik}(\Omega) \frac{1}{1+e^{-\omega_k}} - c_i \right)^2 \tag{J.9}$$

Les fonctions $\alpha_{ik}(\Omega)$ ne sont pas dérivable par rapport Ω mais il est possible d'effectuer une optimisation sans contrainte par descente de gradient. Les coûts sont donc appris en deux étapes :

Algorithme J.4.1 : apprentissage d'une distance d'édition
Les notations sont celles utilisés pour l'équation (J.9). Les coûts Ω sont tirés aléatoirement.

Etape A : estimation
Les coefficients $\alpha_{ik}(\Omega)$ sont calculées.

Etape B : calcul du gradient
Dans cette étape, les coefficients $\alpha_{ik}(\Omega)$ restent constants. Il suffit alors de minimiser la fonction dérivable $E(\Omega)$ sur $\mathbb{R}^n$, ceci peut être effectué au moyen d'un algorithme de descente de gradient [a] similaire à ceux utilisés pour les réseaux de neurones.

a. Annexes : voir paragraphe C.3, page 206

Etape C : calcul du gradient
Tant que l'erreur $E(\Omega)$ ne converge pas, retour à l'étape A.

Remarque J.4.2: décroissance de l'erreur
A partir du moment où l'étape B de l'algorithme J.4.1 fait décroître l'erreur E, l'erreur E diminue jusqu'à converger puisque l'étape A, qui réestime les coefficients $\alpha_{ik}(\Omega)$, les minimise à $\Omega = (\omega_1, \ldots, \omega_n)$ constant.

Annexe K

Recherche des plus proches voisins

Chercher des mots identiques ou similaires dans un dictionnaire est un problème classique et peut être défini pour tout espace métrique. Soit (E, d) un espace métrique quelconque et $D \subset E$ un ensemble fini quelconque, $x \in E$ est un élément de E et $s \in \mathbb{R}_+$ un réel positif. L'objectif est de trouver le sous-ensemble $D'(x, s) \subset D$ des voisins les plus proches de x tels que :

$$D'(x, s) = \{y \in D \mid d(x, y) \leqslant s\}$$

Afin de déterminer les voisins de x, une méthode simple consiste à estimer toutes les distances entre x et les éléments de D. Le coût de cette méthode est proportionnel au nombre d'éléments de D et à la complexité du calcul de la distance. On distingue généralement deux directions afin d'améliorer la rapidité des algorithmes de recherche :

1. L'optimisation du calcul de la distance.
2. L'optimisation de la recherche des voisins.

Les méthodes présentées dans ce chapitre concerne la seconde direction, plus générale que la première.

K.1 Classification ascendante hiérarchique

Cette méthode reprend celle décrite dans l'article [Dupré2003].

K.1.1 Arbre de partitionnement

L'objectif de cette partie est de construire un arbre de partitionnement qui sera utilisé ensuite afin d'améliorer la recherche des plus proches voisins au paragraphe K.1.3.

Définition K.1.1 : rayon et centre d'un ensemble discret
Soit $D = (y_1, \ldots, y_N) \subset E$ un ensemble fini de E, le centre $C(D)$ de D est défini par :

$$C(D) \in \arg\min_{x \in D} \left[\max_{y \in D} d(x, y) \right]$$

où $d(x, y)$ est la distance entre les éléments x et y. On définit aussi le rayon $R(D)$ de D par :

$$R(D) = \max_{x \in D} d(C(D), y)$$

Remarque K.1.2: cas particulier
Si $A, B \subset D$ et $A \subset B$, cela n'implique pas que $R(A) \leqslant R(B)$ comme le montre la figure K.1 où

$$R(A) = d(x, y) > d(x, z) = R(B)$$

.

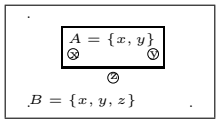


Fig. K.1: Exemple où ajouter un élément à un sous-ensemble aboutit à une réduction du rayon.

Propriété K.1.3 : rayon d'un couple d'éléments
Soit $(x, y) \in E^2$ deux éléments de E, alors $R\{x, y\} = d(x, y)$.

L'algorithme K.1.5 [1] est basé sur une classification ascendante hiérarchique (voir [Saporta1990],

1. Autre formulation :

Algorithme K.1.4 : Arbre de partionnement
Soit $D = (y_1, \ldots, y_N)$ un ensemble fini de (E, d). Soit $N(n_1, n_2, C, R)$ un nœud lié à ses deux précédesseurs n_1, n_2 et qui définit une partie dont le centre est C et le rayon R. L représente un ensemble de nœuds. Si x est un nœud, $P(x)$ désigne la partie réunion de tous les centres des ancêtres de x.
Etape A : initialisation
Pour t$y \in D$, on ajoute le nœud$N(\emptyset, \emptyset, y, 0)$ à L.

Etape B : recherche de la meilleure réunion
Soit $(x, y) \in \arg\min_{x,y \in L,\, x \neq y} R(P(x) \cup P(y))$. Le nœud
$z = N(x, y, C(P(x) \cup P(y)), R(P(x) \cup P(y)))$ est créé et $L \longleftarrow L \cup z - \{x, y\}$

Etape C : terminaison
Si L contient plus d'un nœud, retour à l'étape B.

[Reinert1979]), il construit une hiérarchie de partitions décrite par un graphe.

Algorithme K.1.5 : classification ascendante hiérarchique
Soit $D = (y_1, \ldots, y_N)$ un sous-ensemble fini de (E, d). On note P_n une partie contenant les éléments $P_n = \left\{p_{n,1}\ , ...,\ p_{n,[card(D)-n+1]}\right\}$. L'algorithme a pour but de construire la suite de partitions $(P_n)_{n\geqslant 1}$ comme suit :
Etape A : initialisation
$n = 1$ et P_1 est la partition où chaque élément de D est un élément de P_1

Etape B : récurrence
pour $n = 1$ **à** $N - 1$ **faire**
soit $(i_n^*, j_n^*) \in \underset{i\neq j}{\arg\min}\, R\left(p_{ni} \cup p_{nj}\right)$, alors $P_{n+1} = \left\{(p_{nk})_{k\neq i_n^*, j_n^*}\ , p_{ni^*} \cup p_{nj^*}\right\}$
fin pour

La suite $(P_n)_{1\leqslant n\leqslant N}$ définit un graphe d'inclusion illustré par la figure K.2) pour cinq éléments.

Remarque K.1.6: plusieurs minima
L'étape B impose de choisir un élément dans un ensemble de minima. Lorsque celui-ci contient plus d'un élément, une règle simple consiste à choisir le plus petit regroupement de deux parties parmi celles de rayon minimum. Cette règle a peu d'influence lorsque la construction de l'arbre s'effectue dans un espace continu. En revanche, pour un espace de mot muni d'une distance d'édition comme celle de Levenstein (voir [Levenstein1966]), ce cas se produit fréquemment puisque la distance est à valeurs entières. Cette règle accroît sensiblement les performances.

Chaque nœud du graphe obtenu avec l'algorithme K.1.5 satisfait les conditions suivantes :

1. Il n'a aucun prédécesseur et la partie désignée par ce nœud est un singleton dont le rayon est nul.
2. Il a deux prédécesseurs et la partie pointée par ce nœud contient plus d'un élément, son rayon est strictement positif si au moins deux éléments sont différents.
3. Il n'a aucun successeur et la partie que le nœud désigne est le sous-ensemble D.

Propriété K.1.7 : nombre de nœuds
L'arbre construit par l'algorithme K.1.5 contient exactement $2N - 1 = 2 * card\,(D) - 1$ nœuds.

Démonstration (propriété K.1.7) :
L'arbre construit par l'algorithme K.1.5 contient N nœuds sans prédécesseur, soit un nœud par mot de D. A chaque itération, un nœud est créé pour assembler deux parties entre elles. $N - 1$ itérations sont nécessaires pour passer de N parties à une seule. Donc, le nombre de nœud de l'arbre est :

$$N + (N - 1) = 2N - 1$$

(K.1.7) □

Le théorème K.1.8 montre que l'arbre construit par l'algorithme K.1.5 peut être considéré comme une

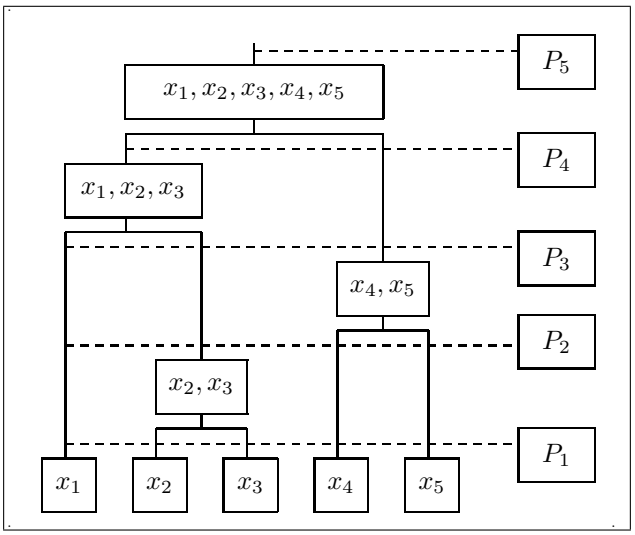


Fig. K.2: Graphe d'inclusion, la première partition contient une partie par élément de D (les feuilles), la seconde partition regroupe ensemble les plus proches éléments, la troisième partition regroupe ensemble les couples d'éléments les plus proches, la quatrième partition construit la partition de rayon minimum, le choix est entre $\{x_1, x_2, x_3\}$, $\{x_1, x_4, x_5\}$, $\{x_2, x_3, x_4, x_5\}$, la cinquième partition est l'ensemble D complet.

hiérarchie pour un certain indice défini par le théorème.

Théorème K.1.8 : hiérarchie
Soit $(P_n)_{1 \leqslant n \leqslant N}$ la suite construite par l'algorithme K.1.5. Soit $i(P_n)$ défini par :

$$i(P_n) = \max_{p \in P_n} R(p)$$

Alors, la suite $(i(P_n))_{1 \leqslant n \leqslant N}$ est croissante.

Démonstration (théorème K.1.8) :

Afin que la démonstration soit plus claire, les partitions notées :

$$(P_{ni})_{\substack{1 \leqslant n \leqslant N \\ 1 \leqslant i \leqslant N-n+1}} \quad \text{avec } \forall (n,i),\ P_{ni} \neq \emptyset$$

sont maintenant notées :

$$(P_{ni})_{\substack{1 \leqslant n \leqslant N \\ 1 \leqslant i \leqslant N}} \text{ avec } \forall n,\ card\{i | P_{ni} = \emptyset\} = N - n + 1$$

Donc :

$$P_1 = \left(P_1^1, \ldots, P_1^N\right) = (\{y_1\}, \ldots, \{y_N\}) \text{ et } P_k = \left(P_k^1, \ldots, P_k^N\right)$$

Selon l'algorithme K.1.5, $\exists (a_{k+1}, b_{k+1}) \in \{1, \cdots, N\}^2$ tel que $a_{k+1} < b_{k+1}$ et :

$$\begin{array}{l} P_{k+1}^{a_{k+1}} = P_k^{a_{k+1}} \cup P_k^{b_{k+1}}, \quad P_{k+1}^{b_{k+1}} = \emptyset, \quad P_k^{a_{k+1}} \neq \emptyset, \quad P_k^{b_{k+1}} \neq \emptyset \\ \text{et } \forall i \notin \{a_{k+1}, b_{k+1}\},\ P_{k+1}^i = P_k^i \end{array}$$

Soit $R_k^i = R\left(P_k^i\right)$ et $C_k^i = C\left(P_i^k\right)$. On impose aussi que si $P_i^k = \emptyset$ alors $R_k^i = 0$. Si l'assertion (K.1) est vraie alors le théorème K.1.8 le sera aussi :

$$\forall k \in \{1, \cdots, N-1\}, \forall i \in \{1, \cdots, N\}, R_{k+1}^{a_{k+1}} \geqslant R_k^i \tag{K.1}$$

Cette démonstratin s'effectue par récurrence. L'assertion (K.1) est vraie de manière évidente pour $k = 2$ puisque :

$$\forall i \in \{1, \cdots, N\}, R_1^i = 0$$

Donc, on suppose (K.1) est pour tout $k < N$, on cherche à montrer que (K.1) est aussi vraie pour $k+1$.

Partie A (démonstration de K.1.8) 1° **cas :** $\{a_k, b_k\} \bigcap \{a_{k+1}, b_{k+1}\} = \emptyset$

On a :

$$\{a_k, b_k\} \bigcap \{a_{k+1}, b_{k+1}\} = \emptyset \Longrightarrow P_{k+1}^{a_k} = P_k^{a_k}$$

Donc :

$$\forall i \in \{1, \cdots, N\}, \quad i \notin \{a_k, b_k\} \Longrightarrow R_k^{a_k} \geqslant R_{k-1}^i = R_k^i \tag{K.2}$$

Mais l'algorithme K.1.8 implique que :

$$\{a_k, b_k\} \in \underset{\substack{i<j \\ P_{k-1}^i \neq \emptyset \\ P_{k-1}^j \neq \emptyset}}{\arg\min} R\left(P_{k-1}^i \bigcup P_{k-1}^j\right) \text{ et } \{a_{k+1}, b_{k+1}\} \in \underset{\substack{i<j \\ P_k^i \neq \emptyset \\ P_k^j \neq \emptyset}}{\arg\min} R\left(P_k^i \bigcup P_k^j\right)$$

Par conséquent, $\{a_k, b_k\} \bigcap \{a_{k+1}, b_{k+1}\} = \emptyset$ implique que :

$$\begin{aligned}
\{a_{k+1}, b_{k+1}\} \in \underset{\substack{i<j \\ i \neq a_k, j \neq b_k \\ P_k^i \neq \emptyset \\ P_k^j \neq \emptyset}}{\arg\min} R\left(P_k^i \bigcup P_k^j\right) &\Longrightarrow \{a_{k+1}, b_{k+1}\} \in \underset{\substack{i<j \\ i \neq a_k, j \neq b_k \\ P_{k-1}^i \neq \emptyset \\ P_{k-1}^j \neq \emptyset}}{\arg\min} R\left(P_{k-1}^i \bigcup P_{k-1}^j\right) \\
&\Longrightarrow R\left(P_k^{a_{k-1}} \bigcup P_k^{b_{k-1}}\right) \leqslant R\left(P_k^{a_{k+1}} \bigcup P_k^{b_{k+1}}\right) \\
&\Longrightarrow R\left(P_k^{a_k}\right) \leqslant R\left(P_{k+1}^{a_{k+1}}\right)
\end{aligned}$$

D'où :

$$\forall i \in \{1, \cdots, N\}, R_{k+1}^{a_k} \geqslant R_k^i \tag{K.3}$$

Partie B (démonstration de K.1.8) 2° **cas :** $\{a_k, b_k\} \bigcap \{a_{k+1}, b_{k+1}\} \neq \emptyset$

Si $\{a_k, b_k\} \bigcap \{a_{k+1}, b_{k+1}\} \neq \emptyset$, alors $a_{k+1} = a_k$ or $b_{k+1} = a_k$. Pour ces deux cas, la preuve est la même donc on suppose que $a_{k+1} = a_k$, alors :

$$\forall i \in \{1, \cdots, N\}, R_k^{a_k} \geqslant R_k^i \quad \text{(voir (K.2))}$$

Comme $a_{k+1} = a_k$, il suffit de prouver que :

$$R_{k+1}^{a_k} \geqslant R_k^{a_k}$$

Mais :

$$P_{k+1}^{a_{k+1}} = P_k^{a_{k+1}} \bigcup P_k^{b_{k+1}} = P_{k-1}^{a_k} \bigcup P_{k-1}^{b_k} \bigcup P_k^{b_{k+1}}$$

Deux cas sont possibles :

1. si $C_{k+1}^{a_k} \in P_k^{a_k}$, alors :

$$R_k^{a_k} = \max_{y \in P_k^{a_k}} d\left(C_k^{a_k}, y\right) \leqslant \max_{y \in P_k^{a_k}} d\left(C_{k+1}^{a_k}, y\right) \leqslant \max_{y \in P_{k+1}^{a_k}} d\left(C_{k+1}^{a_k}, y\right) \leqslant R_{k+1}^{a_k} \tag{K.4}$$

2. si $C_{k+1}^{a_k} \in P_k^{b_{k+1}}$, alors l'algorithm K.1.5 implique que :

$$\begin{aligned} R_{k+1}^{a_k} &= \max \left\{ \max_{y \in P_k^{a_k}} d\left(C_{k+1}^{a_k}, y\right), \max_{y \in P_k^{b_{k+1}}} d\left(C_{k+1}^{a_k}, y\right) \right\} \\ &\geqslant \max \left\{ \max_{y \in P_k^{a_k}} d\left(C_{k+1}^{a_k}, y\right), R_k^{b_{k+1}} \right\} \\ &\geqslant \max \left\{ \max_{y \in P_{k-1}^{a_k}} d\left(C_{k+1}^{a_k}, y\right), \max_{y \in P_{k-1}^{b_k}} d\left(C_{k+1}^{a_k}, y\right), R_k^{b_{k+1}} \right\} \\ &\geqslant \max \left\{ R_k^{a_k}, R_k^{b_k}, R_k^{b_{k+1}} \right\} \\ &\geqslant R_k^{a_k} \end{aligned} \tag{K.5}$$

En conclusion, la récurrence est démontrée par les inégalités (K.3), (K.4), (K.5). Par conséquent, la suite d'indices $(i(P_n))_{1 \leqslant n \leqslant N}$ est croissante.

(K.1.8) □

Corollaire K.1.9 : majoration du rayon
Soit D un sous-ensemble fini de E et A l'arbre obtenu grâce à l'algorithme K.1.5, soit n un nœud quelconque de cet arbre, alors :

$$\text{le successeur } s(n) \text{ de } n \textit{ existe} \Longrightarrow R(n) \leqslant R(s(n)) \tag{K.6}$$

Finalement, si p_1 et p_2 sont deux partitions de l'arbre de partitionnement, et $p_1 \subset p_2$, alors $R(p_1) \leqslant R(p_2)$. Cette conclusion n'était pas évidente d'après le cas particulier de la figure K.1.

L'objectif de cet algorithme est de grouper ensemble les éléments ou les parties les plus proches à chaque itération. La conséquence attendue est que le voisinage d'un mot soit concentré dans une branche de l'arbre. Néanmoins, le principal inconvénient de cet algorithme est son coût. Si on suppose que le coût de la distance est approchée par une constante c et que l'ensemble à hiérarchiser contient n éléments, le coût de l'algorithme est en $O\left(c\, n^5\right)$. Ce coût peut être réduit en factorisant les calculs d'une itération à l'autre puisque la liste L conserve $n - k - 1$ nœuds inchangés. Cette remarque permet de ne calculer le centre et le rayon que pour les parties nouvellement créées. De la même manière, il est possible de conserver pour

chaque nœud, le meilleur voisin qui, à l'itération suivante, peut être resté le même ou être la partie qui vient d'être réunie. Finalement, il est possible de faire descendre le coût de l'algorithme à $O\left(c\,n^3\right)$.

Toutefois, pour des ensembles de plusieurs milliers d'éléments, l'algorithme K.1.5 demeure très long. Une optimisation consiste à adapter l'algorithme des centres mobiles [2]. L'algorithme s'inspire de l'algorithme H.1.1 en remplaçant toutefois la notion de barycentre d'une partie par son centre (définition K.1.1) et l'inertie d'une partie par son rayon.

K.1.2 Ajouter un élément au graphe

L'algorithme K.1.5 ne permet d'insérer de nouveaux nœuds une fois que celui-ci est construit, inconvénient auquel remédie l'algorithme K.1.10. Ce dernier ajoute des nœuds au graphe de partitionnement et pourrait être également utilisé pour construire l'arbre entier en insérant un à un tous les éléments de D mais cette

2. Annexes : voir paragraphe H.1, page 346

méthode est moins efficace.

Algorithme K.1.10 : insertion d'un nœud
Soit D un sous-ensemble fini de E et A un arbre binaire, soit n un nœquelconque de A, alors :

1. n définit une partie notée $P(n)$ dont le rayon est $R(n)$ et dont le centre est $C(n)$.
2. n n'a pas de prédécesseur si $P(n)$ est un singleton ou deux predecessors sinon. Dans ce second cas, $P(n)$ est la réunion des parties de deux prédécesseurs : $Pr(n) = \{p_1(n), p_2(n)\}$
3. n n'a pas de successeur et il définit la partiet D ou un successeur noté $Su(n) = s(n)$

Le seul nœud sans successeur est appelé *racine.* et noté r. Soit m un élément et x le nœud associé, alors $P(x) = \{m\}$, $C(x) = m$, $R(x) = 0$, et x n'a pour le moment aucun succeseur et aucun prédécesseur. N définit un ensemble de nœud, le nœud x est inséré dans l'arbre A selon les règles suivantes :

Etape A : intialisation
$N \longleftarrow \{r\}$

Etape B : insertion
tant que (non fin) faire
si $d(m, C(n)) > R(n)$ (K.7)
alors

1. soit $n \in \arg\min \{d(m, C(n')) \mid n' \in N\}$, le nœud y est créé
2. le successeur de y devient : $s(y) \longleftarrow s(n)$
3. le prédécesseur de y devient : $p_1(y) \longleftarrow x$ and $p_2(y) \longleftarrow n$
4. si le successeur de n existe alors :

$$\exists i \in \{1, 2\} \text{ tel que } p_i(s(n)) = n \text{ et } p_i(s(n)) \longleftarrow p_i(s(n)) = y$$

5. le successeur de n devient : $s(n) \longleftarrow y$
6. le successeur de x devient : $s(x) \longleftarrow y$
7. si $r = n$, alors $r \longleftarrow y$
8. le centre et le rayon sont reestimés pour toutes les parties définies par les éléments $(n_1, ...)$ de la suite the serie :

$$n_0 = n \text{ et pour } k \geqslant 0,\ n_{k+1} = \begin{cases} s(n_k) \text{ si } r \neq n_k \\ r \text{ sinon} \end{cases}$$

9. fin

sinon

$\exists n \in N$ tel que $d(m, C(n)) \leqslant R(n)$ (K.8)
et $N \longleftarrow \{N - \{n\}\} \cup \{p_1(n), p_2(n)\}$

fin si
fin tant que

Remarque K.1.11: hiérarchie
Il n'est pas démontré que l'arbre obtenu après une ou plusieurs applications de l'algorithme K.1.10 vérifie le corollaire K.1.9.

Remarque K.1.12: ordre d'insertion
L'arbre final dépend de l'ordre d'insertion des éléments comme le montre la figure K.3. L'algorithme K.1.10

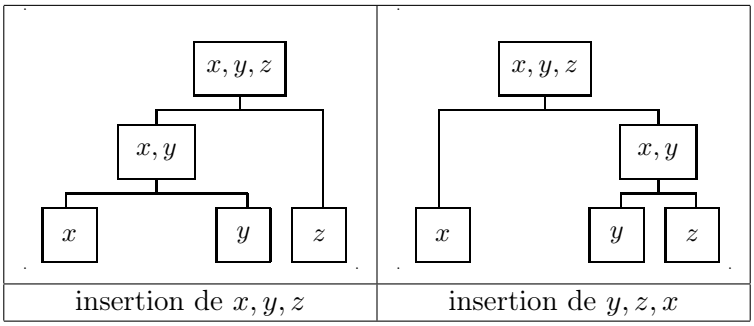


Fig. K.3: Graphe d'inclusion pour deux ordres différents d'inclusion, le second est bien sûr meilleur.

peut être amélioré et devenir l'algorithme $K.1.10^*$ en remplaçant les lignes (K.7) et (K.8) par les suivantes, respectivement (K.9) et (K.10) :

$$\text{si} \quad \forall n \in N,\ d\left(m, ArgC\left(n\right)\right) > R\left(n\right) \tag{K.9}$$

$$\text{sinon} \quad \exists n \in N \text{ tel que } d\left(m, ArgC\left(n\right)\right) \leqslant R\left(n\right) \tag{K.10}$$

où :

$$ArgC\left(n\right) = \underset{x \in P(n)}{\arg\min} \left[\max_{y \in P(n)} d\left(x, y\right) \right] \text{ et } d\left(m, ArgC\left(n\right)\right) = \min_{y \in ArgC(n)} d\left(m, y\right)$$

On ne considère pas seulement un centre mais l'ensemble des centres possibles, à égale distance de l'élément à insérer. Comme la distance de Levenstein est à valeurs entières, cet ensemble est rarement réduit à un singleton comme le montrera le paragraphe K.1.5. Cette version de l'algorithme K.1.10 est notée $K.1.10^*$. La construction de l'arbre de partitionnement est effectuée par la répétition de l'algorithme K.1.10 ou $K.1.10^*$ tant qu'il reste des éléments à classer.

K.1.3 Optimisation de la recherche des plus proches voisins

Cette optimisation de la recherche utilise un des arbres construits par l'algorithme K.1.5 ou la répétition de l'algorithme K.1.10 ou $K.1.10^*$. Chaque nœud définit une partie décrite par un centre et un rayon. Le problème à résoudre consiste ici à trouver pour un élément m la liste $B\left(s\right)$ des voisins inclus dans le sous-ensemble $D\left(y_1, \ldots, y_N\right)$ vérifiant :

$$B\left(s\right) = \left\{x \in D \mid d\left(x, m\right) \leqslant s\right\}$$

Soit $P \subset E$ une partie dont le centre est $C\left(P\right)$ et le rayon $R\left(P\right)$, l'optimisation est basée sur les deux remarques suivantes :

$$d\left(m, C\left(P\right)\right) > s + R\left(P\right) \Longrightarrow \forall w \in P,\ d\left(m, w\right) > s \Longrightarrow B\left(s\right) \cap P = \emptyset \tag{K.11}$$

$$d\left(m, C\left(P\right)\right) + R\left(P\right) \leqslant s \Longrightarrow \forall w \in P,\ d\left(m, w\right) \leqslant s \Longrightarrow B\left(s\right) \subset P \tag{K.12}$$

Algorithme K.1.13 : recherche rapide
Soit r la racine de l'arbre A obtenu par un des algorithmes K.1.5, **??**, $K.1.10^*$. N est un ensemble de nœuds. Soit $s \in \mathbb{R}_+$, $B(s)$ est l'ensemble cherché, il est défini par $B(s) = \{x \in D \mid d(x,m) \leqslant s\}$.

Etape A : initialisation
$N \longleftarrow r$
$B(s) \longleftarrow \emptyset$

Etape B : suite
tant que $(N \neq \emptyset)$ **faire**
Soit $n \in N$ et p la partie définie par n, n est retiré de N : $N \longleftarrow N \backslash n$ et :
si $d(m, C(p)) + R(p) \leqslant s$ **alors**
$B(s) \longleftarrow B(s) \bigcup p$
sinon si $d(m, C(p)) > s + R(p)$
ne rien faire
sinon
si $p = \{w \in D\}$ **alors**
si $d(m,w) \leqslant s$ **alors**
$B(s) \longleftarrow B(s) \bigcup \{w\}$
fin si
sinon
$N \longleftarrow N \bigcup \{p_1(n), p_2(n)\}$
où $\{p_1(n), p_1(n)\}$ sont les deux prédécesseurs de n
fin si
fin si
fin tant que

La liste $B(s)$ contient tous les voisins de m sans aucune approximation.

Remarque K.1.14: mémorisation des distances
Durant l'étape B de l'algorithme K.1.13, il est nécessaire de calculer les distance entre l'élément m et le centre de certaines parties. Ces centres appartiennent au sous-ensemble D et plusieurs parties peuvent avoir le même centre si elles sont incluses les unes dans les autres. Si le calcul de la distance d est coûteux, il est intéressant de conserver en mémoire les résultats du calcul des distances de m aux centres visités. Cette mémorisation implique qu'il ne peut y avoir plus de N calculs de distance si N est le nombre d'éléments de D.

K.1.4 Critère d'efficacité

Quelque soit la méthode choisie pour construire l'arbre de partitionnement, l'algorithme K.1.13 mène à la solution exacte. D'un autre côté, on peut se demander s'il existe des arbres meilleurs que d'autres lors de la recherche des plus proches voisins et s'il existe un arbre optimal. La première idée est de considérer que plus les parties définies par l'arbre sont petites, plus la recherche sera rapide. Selon cette idée, le critère (K.13) essaye d'évaluer la pertinence d'un arbre A construit à partir du sous-ensemble fini D :

$$Cr_1(A) = \begin{cases} 0 \text{ si } R(D) = 0 \text{ ou } card(D) \leqslant 1 \\ \\ \text{sinon } \dfrac{\sum\limits_{n \in A} R(n)}{(card(D) - 1) * R(D)} \end{cases} \tag{K.13}$$

où

$card(D)$	est le nombre d'éléments de D
$R(D)$	est le rayon de D
n	est un nœud de A
$R(n)$	est le centre de la partie définie par n

Si l'arbre A choisi est construit par l'algorithme K.1.5, le corollaire K.1.9 permet d'affirmer que si n est un nœud de A, alors $R(n) \leqslant R(D)$. De plus, l'arbre contient au plus $card(D) - 1$ nœuds dont le rayon est strictement positif :

$$card\{n \in A \mid R(n) > 0\} \leqslant card(D) - 1$$

Par conséquent, l'arbre A construit par l'algorithme K.1.5 satisfait :

$$R(D) \leqslant \sum_{n \in A} R(n) \leqslant (card(D) - 1) * R(D) \tag{K.14}$$

L'inégalité (K.14) explique l'expression du critère (K.13) puisque :

$$R(D) > 0 \Longrightarrow \frac{1}{card(D) - 1} \leqslant Cr_1(A) \leqslant 1 \tag{K.15}$$

Remarque K.1.15: limites
Les deux limites de l'inégalité (K.15) sont atteintes pour un sous-ensemble D ne contenant que deux éléments distincts. Pour un ensemble contenant plus de trois éléments, la propriété suivante répond par-

tiellement à la question.

Propriété K.1.16 : limites
Soit (E, d) un espace métrique quelconque, soit $D \neq \emptyset$ un sous-ensemble fini de E, soit A_D l'arbre construit par l'algorithme K.1.5, alors les trois propositions suivantes sont vraies :

(1) $\forall n > 1$, il existe $D_1 \subset E$ tel que :

$$\left\{ \begin{array}{rcl} card\,(D_1) & = & n \\ Cr_1\,(A_{D_1}) & = & \dfrac{1}{n-1} \end{array} \right.$$

(2) $\forall n > 1$, il existe $D_2 \subset E$ tel que :

$$\left\{ \begin{array}{l} card\,(D_2) = n \\ \forall\,(x,y) \in D_2^2,\ x \neq y \Longrightarrow d\,(x,y) = R\,(D_2) \end{array} \right.$$

alors $Cr_1\,(A_{D_2}) = 1$.

(3) si E est un espace vectoriel de dimension infinie, alors, $\forall n > 1$, il existe $D_3 \subset E$ tel que :

$$\left\{ \begin{array}{rcl} card\,(D_3) & = & n \\ Cr_1\,(A_{D_3}) & = & 1 \end{array} \right.$$

Démonstration (propriété K.1.16) :

Pour prouver **(1)**, il faut considérer l'ensemble $D_1 = (x_1, \ldots, x_n)$ défini par :

$$\left\{ \begin{array}{l} x_1 \neq x_2 \\ \forall i \geqslant 3,\ x_i = x_2 \end{array} \right.$$

De manière évidente : $Cr_1\,(A_{D_1}) = \frac{1}{n-1}$.

Prouver **(2)** est aussi évident parce que si un tel ensemble D_2 existe, pour une partie P quelconque de D, $R\,(P) = R\,(D_2) = Cr_1\,(A)$.

Pour prouver **(3)**, on n'utilise **(2)** puisqu'un tel ensemble D_2 existe dans un espace vectoriel de dimension infinie. C'est précisément le cas de l'espace des mots.

(K.1.16) □

K.1.5 Résultats expérimentaux

La première expérience consiste à chercher les voisins dans un ensemble de points tirés aléatoirement dans le carré $[0,1] \times [0,1]$ (figure K.4). L'expérience consiste d'abord à tirer N points aléatoires dans ce carré. Pour différentes valeurs de seuil s, N points sont de nouveau tirés aléatoirement pour lesquels le voisinage $V_s\,(x) = \{y \mid d\,(x,y) \leqslant s\}$ est calculé selon les deux algorithmes K.1.5 et $K.1.10^*$. Si X est une variable aléatoire de l'espace métrique E - les éléments de E sont équiprobables -, l'objectif est d'estimer le nombre moyen de calculs de distance $r_s\,(N)$ effectués pour déterminer les voisins d'un élément :

$$r_s\,(N) = \frac{1}{N}\,\mathbb{E}\,(\text{nombre de distances calculées pour } V_s\,(X)) \tag{K.16}$$

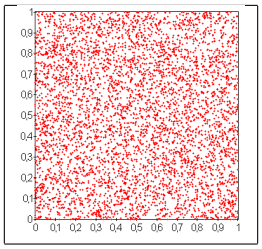


Fig. K.4: Tirage aléatoire de points dans le carré $[0,1] \times [0,1]$.

L'algorithme K.1.5 est de loin le meilleur et ce quelle que soit la valeur du seuil s choisi. Cette supériorité est également traduite par la valeur de Cr_1 obtenu pour chacun des arbres (voir table K.1).

seuil	$\frac{\mathbb{E}(card(V_s(X)))}{N}$	$r_s(N{=}2000)$ *algorithme* $K.1.10^*$	$r_s(N{=}2000)$ *algorithme* $K.1.5$	$r_s(N{=}5000)$ *algorithme* $K.1.5$	$r_s(N{=}10000)$ *algorithme* $K.1.5$
0,001	0 %	6,4 %	1,2 %	0,6 %	0,3 %
0,01	0 %	6,8 %	1,4 %	0,7 %	0,4 %
0,1	2 %	11,8 %	4,1 %	2,7 %	1,9 %
0,2	10 %	17,2 %	6,8 %	4,5 %	3,2 %
0,3	21 %	21,7 %	8,7 %	5,7 %	4,0 %
0,4	34 %	25,1 %	9,6 %	6,3 %	4,5 %
0,5	48 %	26,9 %	10,0 %	6,5 %	4,6 %
0,6	62 %	27,9 %	9,4 %	6,1 %	4,4 %
0,7	74 %	27,0 %	8,3 %	5,4 %	3,9 %
0,8	85 %	24,0 %	6,7 %	4,3 %	3,1 %
0,9	92 %	19,2 %	4,6 %	3,0 %	2,1 %
1	98 %	10,9 %	2,3 %	1,4 %	1,0 %
2	100 %	0,1 %	0,1 %	0,0 %	0,0 %
Cr_1	-	0,135	0,039	0,023	0,017
temps de calcul (arbre)	-	$\sim$ 2 sec	$\sim$30 sec	$\sim$2 min	$\sim$10 min

Tab. K.1: Gain apporté lors de la recherche du voisinage pour différentes valeurs de seuil. Le premier test utilise un nuage de 2000 points tirés aléatoirement dans l'ensemble $[0,1]^2$, le second en utilise 5000, le dernier 10000. A seuil fixe, la part du voisinage observé décroît lorsque N diminue. Dans les trois cas, lorsque le seuil est fixé, les rapports tailles de voisinages sur nombre d'éléments sont sensiblement égales quel que soit N. On s'aperçoit que le critère Cr_1 décroît également lorsque N augmente. Les temps de calcul sont estimées avec un processeur Intel Pentium III à 1 GHz et désignent le temps nécessaire à la construction de l'arbre.

Une expérience similaire est effectuée dans un espace de mots et pour mesurer l'amélioration obtenu par l'optimisation décrite au paragraphe K.1.3, le test suivant est réalisé :

– Un dictionnaire D de 2178 prénoms est utilisé, son rayon est 10.
– Le test consiste en l'obtention du voisinage $V_s(m)$ de n'importe quel mot m du dictionnaire.
– La distance utilisée est cette de Levenstein ([Levenstein1966], [Wagner1974]).

Sans optimisation, pour un mot donné m, toutes les distances de m avec les autres mots doivent être calculées. En utilisant l'optimisation proposée, il n'est pas nécessaire de les calculer toutes. Les résultats sont illustrés par le tableau K.2.

Remarque K.1.17: ordre d'insertion
L'ordre d'insertion des mots dans l'arbre affecte les résultats. En ce qui concerne les algorithmes K.1.10 et $K.1.10^*$, les mots ont été insérés par ordres croissant et décroissant de taille, les différences sont rendues

seuil	$\frac{\mathbb{E}(card(V_s(X)))}{N}$	$r_s(N{=}2178)$ *algorithme K.*1.5	$r_s(N{=}2178)$ *algorithme K.*1.10*
1	0,1 %	17,3 %	34,1 %
2	0,3 %	30,2 %	46,9 %
3	1,5 %	45,7 %	60,4 %
4	7,0 %	63,1 %	73,3 %
5	21,8 %	76,3 %	83,0 %
6	45,2 %	79,8 %	86,0 %
7	68,2 %	74,0 %	81,2 %
8	82,8 %	59,6 %	71,1 %
9	90,9 %	45,2 %	58,0 %
10	95,5 %	30,7 %	43,5 %
11	96,7 %	21,0 %	28,8 %
12	98,5 %	12,5 %	18,3 %
13	99,3 %	7,7 %	11,0 %
14	99,4 %	4,9 %	7,3 %
15	99,5 %	3,0 %	4,2 %
16	99,5 %	2,2 %	2,7 %
17	99,6 %	1,5 %	2,0 %
18	99,8 %	0,9 %	1,5 %
19	99,7 %	0,9 %	1,3 %
20	99,9 %	0,6 %	1,2 %
Cr_1	-	0,153	0,221
temps de calcul (arbre)	-	∼5 min	∼1 h

Tab. K.2: Amélioration moyenne mesurée par (K.16), comparaison des algorithmes K.1.5, *K.*1.10*. Le centre du dictionnaire est MARIE-LOUISE et son rayon est 17. Dans le pire des cas, $s = 6$, 20% des calculs de distances sont évités. Les temps de calcul correspondand à la construction de l'arbre sont estimés avec un processeur Intel Pentium 1 GHz.

par le tableau K.3.

K.2 Voisinage dans un espace vectoriel

Lorsque l'espace métrique est aussi vectoriel, la recherche des plus proches voisins est facilité car il est possible d'utiliser les coordonnées des éléments comme dans l'algorithme *Branch and Bound*. Ces coordonnées permettent également d'obtenir des résultats théorique plus avancés en ce qui concerne le coût de cette recherche (voir [Arya1994].

K.2.1 B+ tree

Ce premier algorithme s'applique dans le cas réel afin d'ordonner des nombres dans un arbre de sorte que chaque nœud ait un père et pas plus de n fils (voir figure K.5).

seuil	r moyen *algorithme K.1.10** longueurs décroissantes	r moyen *algorithme K.1.10** longueurs croissantes
1	$82,0\,\%$	$75,2\,\%$
2	$65,8\,\%$	$60,6\,\%$
3	$45,6\,\%$	$42,8\,\%$
4	$26,7\,\%$	$25,5\,\%$
5	$12,8\,\%$	$12,2\,\%$

Tab. K.3: Amélioration moyenne mesurée par (K.16), comparaison des ordres d'insertion

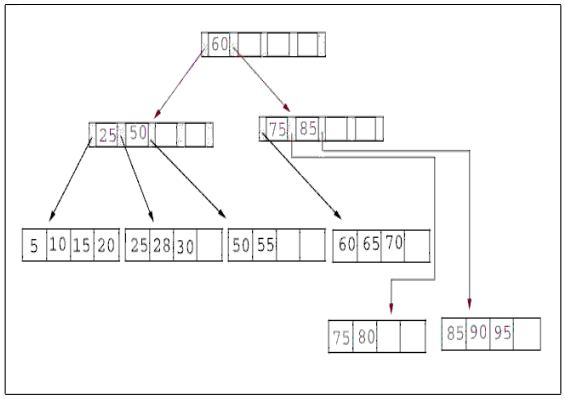


Fig. K.5: Illustration d'un B+ tree.

Définition K.2.1 : B+ tree
Soit B_n un B+ tree, soit N un nœud de B_n, il contient un vecteur $V(N) = (x_1, \ldots, x_t)$ avec $0 \leqslant t \leqslant n$ et $x_1 < \ldots < x_t$. Ce nœud contient aussi exactement $t-1$ nœuds fils notés $(N_1, \ldots, N_{t-1})$. On désigne par $D(N_t)$ l'ensemble des descendants du nœud N_t et $G(N_t) = \{V(M) \mid M \in D(N_t)\}$. Le nœud N vérifie :

$$\forall x \in G(N_t),\ x_t \leqslant x < x_{t+1}$$
avec par convention $x_0 = -\infty$ et $x_{t+1} = +\infty$

Cet arbre permet de trier une liste de nombres, c'est une généralisation du tri "quicksort" pour lequel $n = 2$. Comme pour le tri quicksort, l'arbre est construit à partir d'une série d'insertions et de cet ordre dépend la rapidité du tri. L'espérance du coût (moyenne sur tous les permutations possibles de k éléments), le coût de l'algorithme est en $O(k \log_n k)$.

K.2.2 R-tree ou Rectangular Tree

L'arbre R-tree est l'adaptation du mécanisme du B+ tree au cas multidimensionnel (voir [Guttman1984]). La construction de cet arbre peut se faire de manière globale - construction de l'arbre sachant l'ensemble de points à classer - ou de manière progressive - insertion des points dans l'arbre les uns à la suite des autres -. Ces arbres sont comparables à l'arbre de partitionnement construit par l'algorithme K.1.5 à ceci près que la forme des ensembles est constituée de rectangles et non plus de cercles. L'appartenance d'un point à un rectangle dépend dorénavant des comparaisons entre coordonnées tandis que l'appartenance d'un point à un cercle nécessite le calcul d'une distance, ce qui est plus lent. Toutefois, ces méthodes sont resteintes à des espaces vectoriels.

seuil	$\frac{\mathbb{E}(card(V_s(X)))}{N}$	$r_s(N{=}4987)$ *algorithme K.1.5*
1	0,0 %	5,7 %
2	0,0 %	10,4 %
3	0,0 %	17,1 %
4	0,1 %	26,4 %
5	0,4 %	42,2 %
6	2,0 %	61,6 %
7	9,0 %	82,1 %
8	34,1 %	94,7 %
9	77,4 %	91,3 %
10	97,7 %	73,1 %
11	99,4 %	48,4 %
12	99,9 %	31,0 %
13	100,0 %	20,1 %
14	100,0 %	11,1 %
15	100,0 %	6,0 %
16	100,0 %	3,0 %
17	100,0 %	1,3 %
18	100,0 %	0,0 %
Cr_1	-	0,210
temps de calcul	-	∼3 h

Tab. K.4: Amélioration moyenne mesurée par (K.16), le test est effectué sur un dictionnaire de 4987 mots anglais de centre "POSITIVELY" de rayon 13. L'optimisation est plus pertinente dans ce cas où le dictionnaire contient plus du double de mots que celui utilisé pour le test K.2.

Il n'existe pas une seule manière de construire un R-tree, les nœuds de ces arbres suivent toujours la contrainte des B+ Tree qui est d'avoir un père et au plus n fils. Les R-Tree ont la même structure que les B+ Tree ôtée de leurs contraintes d'ordonnancement des fils. De plus, ces arbres organisent spatialement des rectangles ou boîtes en plusieurs dimensions comme le suggère la figure K.6. Les boîtes à organiser seront nommés les objets, ces objets sont ensuite regroupés dans des boîtes englobantes. Un nœud n d'un R-tree est donc soit une feuille, auquel cas la boîte qu'il désigne est un objet, dans ce cas, il n'a aucun fils, soit le nœud désigne une boîte englobante $B(n)$. On désigne par $\mathcal{B}$ l'ensemble des boîtes d'un espace vectoriel quelconque et $v(b)$ désigne son volume. Pour un nœud n non feuille, $A(n)$ désigne l'ensemble des descendants de ce nœud. $B(n)$ est défini par :

$$B(n) = \arg\min \left\{ v(b) \mid b \in \mathcal{B} \text{ et } \forall n' \in A\left(n'\right),\ B\left(n'\right) \subset B(n) \right\}$$

La recherche dans un R-tree consiste à trouver toutes les objets ayant une intersection avec une autre boîte ou fenêtre W, soit l'ensemble L :

$$L = \{ B(n) \mid B(n) \text{ est un objet et } B(n) \cap W \neq \emptyset \}$$

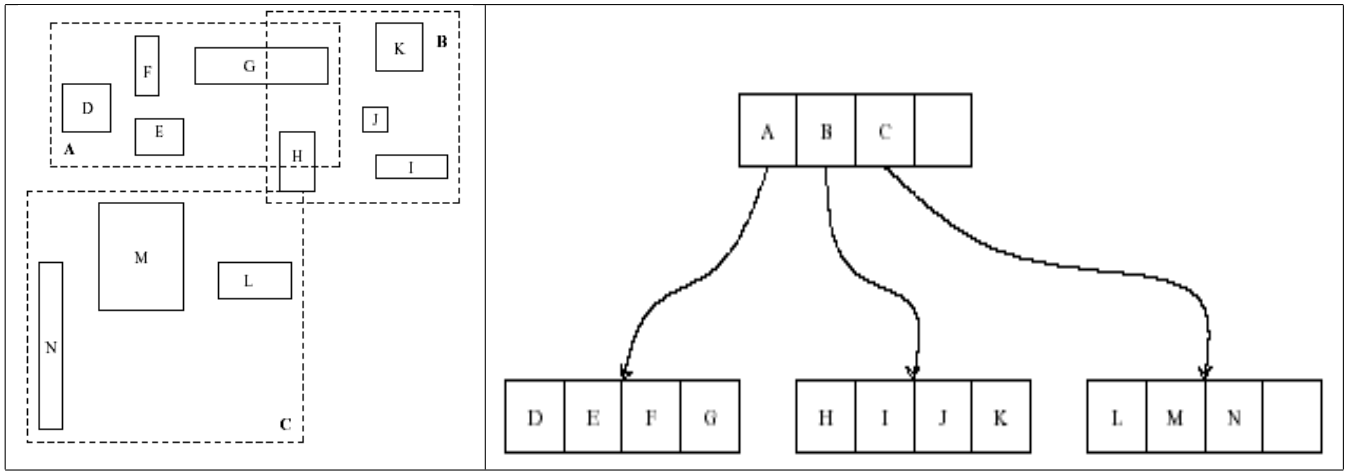


Fig. K.6: Illustration d'un R-tree en deux dimensions, figure extraite de [Sellis1987], la première image montre des rectangles pointillés englobant d'autres rectangles en trait plein. Chaque style de trait correspond à un niveau dans le graphe de la seconde image.

Cet ensemble est construit grâce à l'algorithme suivant :

Algorithme K.2.2 : recherche dans un R-tree
Les notations sont celles utilisées dans ce paragraphe. On désigne par r le nœud racine d'un R-tree. Soit n un nœud, on désigne par $F(n)$ l'ensemble des fils de ce nœud.

Etape A : initialisation
$L \longleftarrow 0$
$N \longleftarrow \{r\}$

Etape B : itération
tant que $(N \neq \emptyset)$ **faire**
pour chaque $n \in N$ **faire**
si $W \cap B(n) \neq \emptyset$ **alors**
$N \longleftarrow N \cup F(n)$
si $B(n)$ **est un objet alors**
$L \longleftarrow B(n)$
fin si
fin si
fin pour
fin tant que

L est l'ensemble cherché.

Il reste à construire le R-tree, opération effectuée par la répétition successive de l'algorithme K.2.3 per-

mettant d'insérer un objet dans un R-tree.

Algorithme K.2.3 : insertion d'un objet dans un R-tree
Les notations utilisées sont les mêmes que celles de l'algorithme K.2.2. On cherche à insérer l'object E désigné par son nœud feuille e. On suppose que l'arbre contient au moins un nœud, sa racine r. On désigne également par $p(n)$ le père du nœud n. Chaque nœud ne peut contenir plus de s fils. On désigne par $v^*(G) = \min \left\{ P \mid P \in \mathcal{B} \text{ et } \bigcup_{g \in G} B(g) \subset P \right\}$.

Etape A : sélection du nœud d'insertion
$n^* \longleftarrow r$
tant que (n^* n'est pas un nœud feuille) faire
On choisit le fils f de n^* qui minimise l'accroissement $v_f - v(B(f))$ du volume avec v_f défini par :

$$v_f = \min \{ v(P) \mid P \in \mathcal{B} \text{ et } B(f) \cup B(e) \subset P \} \tag{K.17}$$

$n^* \longleftarrow f$
fin tant que

Etape B : ajout du nœud
Si $p(n^*)$ a moins de s fils, alors le nœud e devient le fils de $p(n^*)$ et $B(p(n^*))$ est mis à jour d'après l'expression (K.17). L'insertion est terminée. Dans le cas contraire, on sépare découpe le nœud $p(n^*)$ en deux grâce à l'étape suivante.

Etape C : découpage des nœuds
L'objectif est de diviser le groupe G composé de $s+1$ nœuds en deux groupes G_1 et G_1. Tout d'abord, on cherche le couple (n_1, n_2) qui minimise le critère

$$d = v^*(\{n_1, n_2\}) - v(B(n_1)) - v(B(n_2))$$

Alors : $G_1 \longleftarrow n_1$, $G_2 \longleftarrow n_2$ et $G \longleftarrow G - G_1 \cup G_2$
tant que ($G \neq \emptyset$) faire
On choisit un nœud $n \in G$, on détermine i^* tel que $v(\{n\} \cup G_i) - v(G_i)$ soit minimal.
$G \longleftarrow G - \{n\}$
$G_{i^*} \longleftarrow G_{i^*} \cup \{n\}$
fin tant que

Si la recherche est identique quel que soit l'arbre construit, chaque variante de la construction de l'arbre tente de minimiser les intersections des boîtes et leur couverture. Plus précisément, l'étape C qui permet de découper les nœuds est conçue de manière à obtenir des boîtes englobantes de volume minimale et/ou d'intersection minimale avec d'autres boîtes englobantes. L'algorithme R+ Tree (voir [Sellis1987]) essaye de minimiser les intersections entre boîtes et les objets à organiser sont supposés n'avoir aucune intersection commune. La variante R* Tree (voir [Beckmann1990]) effectue un compromis entre l'intersection et la couverture des boîtes englobantes. L'algorithme X-Tree (voir [Berchtold1996]) conserve l'historique de la construction de l'arbre ce qui lui permet de mieux éviter les intersections communes entre boîtes.

K.2.3 Branch and Bound

Les algorithmes regroupés sous cette terminaison *Branch and Bound* englobe la plupart des méthodes présentées dans ce document, elles désignent tout algorithme de recherche nécessitant une première étape permettant de construire une décomposition hiérarchique de l'ensemble des exemples ou exemples d'ap-

prentissage, cet ensemble étant inclus dans un espace vectoriel. La première version de cette famille algorithme a été développée dans [Fukunaga1975].

La méthode présentée dans [D'Haes2003] utilise une analyse en composantes principales afin de construire une hiérarchique adaptée à un nuage de points obéissant à une loi normale multidimensionnelle (voir figure K.7).

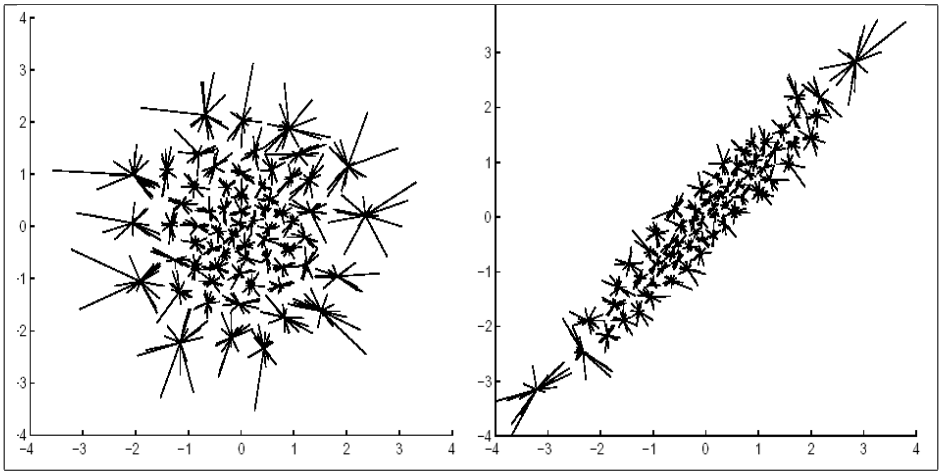


Fig. K.7: Figure extraite de [D'Haes2003] représentant l'arbre de décomposition de deux nuages de points obéissant à des loi normales, uniforme dans le premier cas, avec deux variables corrélées dans le second cas.

La méthode proposée dans cet article effectue une analyse en composantes principales afin de repérer l'axe principal du nuage qui correspond au vecteur propre $\vec{V}$ associé à la plus grande des valeurs propres de la matrice $X'X$ où chaque ligne de la matrice X est un élément du nuage de points $(\vec{x_1}, \ldots, \vec{x_n})$. On détermine la médiane m de l'ensemble $\left\{ < \vec{V}, \vec{x_i} > \mid 1 \leqslant i \leqslant n \right\}$. Le nuage de points est alors divisée en deux sous-nuages de cardinaux égaux selon que le produit scalaire $< \vec{V}, \vec{x_i} >$ est inférieur ou supérieur à m. Chaque sous-nuage est à nouveau divisé selon le même processus incluant une analyse en composantes principales et la recherche de la médiane. L'algorithme s'arrête lorsque les sous-ensembles sont réduits à un seul élément.

K.2.4 Méthodes approchées

Toutes les méthodes proposées jusqu'à présent permettent de déterminer le voisinage exact d'un élément inclus dans un ensemble d'exemples, les différences concernant l'organisation de cet ensemble afin d'optimiser la recherche. L'autre direction de recherche concerne la recherche du voisinage notamment par des méthodes approchées, plus rapide, mais ne retournant pas le voisinage exact. Ces méthodes ne seront pas plus développées ici, l'article [Arya1994] propose une étude théorique de ce problème. L'article [Ramasubramanian2000] compare plusieurs méthodes de recherche du voisinage.

K.3 Autres alternatives

Il existe de nombreuses alternatives en ce qui concerne la recherche des plus proches voisins dans un espace métrique quelconque, passées en revue dans les articles [Bustos2001], [Chavez1999], [Navarro2001], certaines utilisant plus particulièrement les arbres comme [Ulhmann1991] ou [Yianilos1993]. Lorsque cette recherche s'applique aux mots ou aux séquences, l'optimisation de la distance est envisagée (voir [Apostolico1985] ou [Madhvanath2001]). L'algorithme qui suit est une de ces alternatives préférée aux

autres en raison de sa simplicité. Il ne nécessite pas la construction d'un arbre et son coût est aisément calculable.

K.3.1 LAESA

Cet algorithme permet de chercher les plus proches voisins dans un ensemble inclus dans un espace métrique quelconque. Il s'appuie encore sur l'inégalité triangulaire appliquée de manière semblable à (K.11). Le voisinage d'un point x doit être cherché dans un ensemble E. L'algorithme LAESA (Linear Approximating Eliminating Search Algorithm, voir [Rico-Juan2003]) consiste à éviter un trop grand nombre de calculs de distances en se servant de distances déjà calculées entre les éléments de E et un sous-ensemble B inclus dans E contenant des "pivots".

La sélection des pivots demeure un problème ouvert. Ceux-ci pourrait être les nœuds d'un coupe de l'arbre construit par l'algorithme K.1.5. Il existe d'autres possibilités comme l'algorithme K.3.3 plus simple et nettement moins coûteux - les deux algorithmes n'ont pas été comparés en terme de performances en classification.

Algorithme K.3.1 : LAESA
Soit $E = \{y_1, \ldots, y_N\}$ un ensemble de points, $B = \{p_1, \ldots, p_P\} \subset E$ un ensemble de pivots inclus dans E. On cherche à déterminer le voisinage $V(x)$ de x inclus dans E vérifiant :

$$\forall y \in V(x),\ d(x,y) \leqslant \rho$$

On suppose que la matrice $M = (m_{ij})_{\substack{\leqslant i \leqslant P \\ 1 \leqslant j \leqslant N}}$ a été calculée préalablement comme suit :

$$\forall (i,j), m_{ij} = d(p_i, y_j)$$

Etape A : initialisation

$\forall y \in E,\quad g(y) \longleftarrow 0$
$\forall y \in E,\quad h(y) \longleftarrow 1$

Etape B : choix d'un pivot et mise à jour de la fonction g

$B' \longleftarrow B \cap \{y \in E \mid h(y) = 0\}$
si $B' \neq \emptyset$ **alors**
Soit i tel que p_i soit un élément de B' tiré au hasard tel que :

$$p_i \in \arg\min \left\{ g(y) \mid y \in B' \right\}$$

$\alpha \longleftarrow d(p_i, x)$
$h(p_i) \longleftarrow 1$
$\forall y_j \in E$ tel que $h(y_j) = 0,\ g(y_j) \longleftarrow \max\{g(y_j), |\alpha - m_{ij}|\}$
$= \max\{g(y_j), |d(x,p_i) - d(p_i,y_j)|\}$
sinon
Choisir un élément s de E tel que $h(s) = 0$.
$\alpha \longleftarrow d(s,x)$
$h(s) \longleftarrow 1$
$\forall y \in E$ tel que $h(y) = 0,\ g(y) \longleftarrow d(x,y)$
fin si

Etape C : élimination
$\forall y_j \in E$ tel que $h(y_j) = 0$, si $g(y_j) > \rho$ alors $h(y_j) = 1$

Etape D : terminaison
$A \longleftarrow \{y \in E \mid h(y) = 0\}$
si $A \neq \emptyset$ **alors**
Retour à l'étape B.
sinon
Fin, l'ensemble cherché correspond à $\{y \in E \mid g(y)\} \leqslant \rho$.
fin si

La sélection des pivots est assurée par un autre algorithme décrit dans l'article [Moreno2003].

Algorithme K.3.2 : LAESA : sélection des pivots
Soit $E = \{y_1, \ldots, y_N\}$ un ensemble de points, on cherche à déterminer l'ensemble $B = \{p_1, \ldots, p_P\} \subset E$ utilisé par l'algorithme K.3.1.
Etape A : initialisation
$B \longleftarrow y \in E$ choisi arbitrairement.

Etape B : calcul de la fonction g
pour chaque $y \in E - B$ **faire**
$g(y) \longleftarrow 0$
pour chaque $p \in B$ **faire**
$g(y) \longleftarrow g(y) + d(y, p)$
fin pour
fin pour

Etape C : mise à jour de B
Trouver $p^* \in \arg\max \{g(p) \mid p \in E - B\}$
$B \longleftarrow B \cup \{p^*\}$
Si $card(B) < P$, retour à l'étape B sinon fin.

Cet article [Moreno2003] améliore également l'algorithme K.3.1 par le suivant :

Algorithme K.3.3 : LAESA'
Soit $E = \{y_1, \dots, y_N\}$ un ensemble de points, $B = \{p_1, \dots, p_P\} \subset E$ un ensemble de pivots inclus dans E. On cherche à déterminer le voisinage $V(x)$ de x inclus dans E vérifiant :

$$\forall y \in V(x),\ d(x,y) \leqslant \rho$$

On suppose que la matrice $M = (m_{ij})_{\substack{\leqslant i \leqslant P \\ 1 \leqslant j \leqslant N}}$ a été calculée préalablement comme suit :

$$\forall (i,j),\ m_{ij} = d(p_i, y_j)$$

Etape A : initialisation
$\forall i \in \{1, \dots, P\},\ d_i \longleftarrow d(x, p_i)$

Etape B : fonction g
$\forall j \in \{1, \dots, N\},\ g(y_j) \longleftarrow \min_{i \in \{1,\dots,P\}} |m_{ij} - d_i|$

Etape C : tri
Tri l'ensemble $g(y_i)$ par ordre croissant $\longrightarrow g\left(y_{\sigma(j)}\right)$.
pour $j = 1$ **à** N **faire**
  si $g\left(y_{\sigma(j)}\right) \leqslant \rho$ **alors**
    $g\left(y_{\sigma(j)}\right) = d\left(x, y_{\sigma(j)}\right)$
  fin si
fin pour
Fin, l'ensemble cherché correspond à $\{y \in E \mid g(y)\} \leqslant \rho$.

Il existe d'autres versions de l'algorithme LAESA comme TLAESA (Tree - LAESA) (voir [Micó1996]). Cet algorithme associe un arbre à l'algorithme LAESA et fait le lien entre les algorithmes K.1.5 et K.3.1.

K.3.2 Résultats théoriques

L'article [Faragó1993] démontre également qu'il existe une majoration du nombre moyen de calcul de distances pour peu que la mesure de l'espace contenant l'ensemble E et l'élément x soit connue et que l'ensemble $B = \{p_1, \dots, p_P\}$ des pivots vérifie :

$$\exists (\alpha, \beta) \in \mathbb{R}_*^+ \text{ tels que}$$
$$\forall (x,y) \in E^2,\ \forall i \quad \alpha\, d(x,y) \geqslant |d(x,p_i) - d(p_i,y)| \tag{K.18}$$
$$\forall (x,y) \in E^2,\ \max_i\ |d(x,p_i) - d(p_i,y)| \geqslant \beta\, d(x,y) \tag{K.19}$$

L'algorithme développé dans [Faragó1993] permet de trouver le point de plus proche d'un élément x dans

un ensemble $E = \{x_1, \ldots, x_N\}$ selon l'algorithme suivant :

Algorithme K.3.4 : plus proche voisin d'après [Faragó1993]
Soit $E = \{x_1, \ldots, x_N\}$ et $B = \{p_1, \ldots, p_P\} \subset E \subset X$. Soit $x \in X$ un élément quelconque. On suppose que les valeurs $m_{ij} = d(x_i, p_j)$ ont été préalablement calculées.

Etape A : initialisation
On calcule préalablement les coefficients $\gamma(x_i)$:

$$\forall i \in \{1, \ldots, N\},\ \gamma(x_i) \longleftarrow \max_{j \in \{1,\ldots,P\}} |m_{ij} - d(x, p_j)|$$

Etape B : élaguage
On définit $t_0 \longleftarrow \min_i \gamma(x_i)$.
Puis on construit l'ensemble $F(x) = \{x_i \in E \mid \gamma(x_i)\} \leqslant \frac{\alpha}{\beta} t_0$.

Etape C : plus proche voisin
Le plus proche x^* voisin est défini par : $x^* \in \arg\min \{d(x, y) \mid y \in F(x)\}$.

Théorème K.3.5 : [Faragó1993][1]
Les notations sont celles de l'algorithme K.3.4. L'algorithme K.3.4 retourne le plus proche voisin x^* de x inclus dans E. Autrement dit, $\forall x \in X,\ x^* \in F(x)$.

Remarque K.3.6: mesure de dissimilarité
L'algorithme K.3.4 est en fait valable pour une distance mais aussi pour une mesure de dissimilarité. Contrairement à une distance, une mesure de dissimalité ne vérifie pas l'inégalité triangulaire.

Théorème K.3.7 : [Faragó1993][2]
Les notations sont celles de l'algorithme K.3.4. On définit une mesure sur l'ensemble X, $B(x, r)$ désigne la boule de centre x et de rayon r, $Z \in X$ une variable aléatoire, de plus :

$$p(x, r) = P_X(B(x, r)) = \mathbb{P}(Z \in B(x, r))$$

On suppose qu'il existe $d > 0$ et une fonction $f : X \longrightarrow \mathbb{R}$ tels que :

$$\lim_{r \to 0} \frac{p(x, r)}{r^d} = f(x) > 0$$

La convergence doit être uniforme et presque sûre. On note également F_N le nombre de calculs de dissimilarité effectués par l'algorithme K.3.4 où N est le nombre d'élément de E, P désigne toujours le nombre de pivots, alors :

$$\limsup_{n \to \infty} \mathbb{E}(F_N) \leqslant k + \left(\frac{\alpha}{\beta}\right)^{2d}$$

K.3.3 Suppression des voisins inutiles

L'article [Wu2002] propose une idée intéressante qui consiste à supprimer les voisins inutiles. Cette méthode s'applique dans le cas d'une classification à l'aide de plus proches voisins. A un élément à classer, cette

méthode attribue la classe du point le plus proche, il faut donc a priori calculer les distances du point en question à tous ceux déjà classés. Certains points de cet ensemble ont une forte "capacité d'attraction" : ils sont le centre d'une région dans laquelle seuls des points de la même classe figurent. Plus concrètement, soient un point y à classer et une base de points classés noté $(x_1, \ldots, x_N)$ de classe $(c(x_1), \ldots, c(x_N))$, on suppose que x_i et x_j sont deux points, enfin, on définit :

$$y^* = \arg\min \{d(x_k, y) \mid 1 \leqslant k \leqslant n\} \tag{K.20}$$

On suppose également que x_i est un point de forte capacité et que :

$$\forall y,\ y^* = x_i \Longrightarrow \exists l \text{ tel que } x_l = \arg\min \{d(x_k, y) \mid k \neq i\} \text{ et } c(x_l) = c(x_i) \tag{K.21}$$

Autrement dit, x_i est un point attracteur si quel que soit le point y proche de x_i, il sera toujours possible de trouver un voisin de x_i proche et y et appartenant à la même classe. Dans ce cas, il ne sert à rien de calculer la distance de x_i à y, le point x_i peut alors être éliminé de la base des points classés. Il reste maintenant à traiter la base de points classés de manière à en garder le moins possible. De cette façon, l'ensemble des points classés ne gardera que des points situés près des frontières entre classes.

L'article [Wu2002] définit le rayon d'un point :

$$r(x) = \max \{0, \max \{d(x, y) \mid c(x) = c(y)\}\} \tag{K.22}$$

L'algorithme de suppression des points attracteurs est le suivant :

Algorithme K.3.8 : suppression des points attracteurs
Soit Γ un seuil positif, les notations sont celles utilisées dans les paragraphes qui précèdent, on note également Ω l'ensemble des points classés.

Etape A : calcul des rayons
Pour chaque $x \in \Omega$, on calcule $r(x)$, et on désigne par x^* le point qui vérifie :
$r(x^*) = \max_{x \in \Omega} r(x)$

Etape B : suppression
Si $r(x^*) \geqslant \Gamma$, alors le point x^* est supprimé de l'ensemble Ω, et retour à l'étape A. Dans le cas contraire, l'algorithme s'arrête.

Remarque K.3.9: méthode approchée
Le fait de supprimer les points dont le rayon attracteur est supérieur à un certain seuil peut entraîner une modification de la classification si ce seuil est trop petit.

Cette méthode poursuit le même objectif que celui des méthodes LVQ [3] ou Learning Vector Quantization qui permettent de réduire un ensemble de points utilisés dans une classification de plus proches voisins à un ensemble de prototypes.

K.3.4 Lien vers la classification

Déterminer le voisinage d'un point est un passage obligé lorsqu'on applique une classification à l'aide de plus proches voisins puisque chaque élément est classé à partir des classes de ses voisins [4]. Toutefois, les

3. Annexes : voir paragraphe I.4.1, page 375
4. Annexes : voir paragraphe I.1.2, page 374

résultats obtenus par cette méthode dépendent fortement de la distance utilisée. Sans a priori sur celle-ci, c'est souvent une distance euclidienne qui est choisie.

Dans le cas des espaces vectoriels, il est possible d'utiliser une distance pondérant différemment chaque dimension et d'estimer cette pondération à partir d'un échantillon représentatif du problème de classification à résoudre. Deux méthodes sont présentées aux chapitre H.5.2 et H.5.3.

Annexe L

N-grammes

Les *n-grammes* sont une modélisation statistique du langage, l'idée est d'observer la fréquence des enchaînements de lettres à l'intérieur des mots, ou la fréquence des enchaînements de mots à l'intérieur d'une phrase. Plus généralement, les n-grammes modélisent sous forme de chaînes de Markov toute séquence de symboles appartenant à un ensemble fini.

L.1 Définition

Les définitions qui vont suivre s'adaptent à toutes séquences d'éléments appartement à E, un ensemble fini.

Définition L.1.1 : n-grammes
Soit $A = (e, f, a_1, \ldots, a_N)$ un ensemble fini nommé *alphabet*, les symboles e et f débutent et terminent toute séquence de symboles appartenant à A. Cette convention permet de traiter les probabilités d'entrée et de sortie d'une séquence comme des probabilités de transitions. Soit $M_A \subset A^{\mathbb{N}}$ l'ensemble des suites $u = (u_i)_{i \geqslant 0}$ de A définies comme suit :

$$u \in M_A \iff \begin{cases} u_1 = e \\ \exists N > 2 \text{ tel que } \forall i \geqslant N,\ u_i = f \text{ et } \forall i < N,\ u_i \neq f \end{cases}$$

Par la suite, M_A sera appelé l'ensemble des mots de A.

Soit $n \geqslant 2$, M_A est muni d'une distribution de probabilité vérifiant les hypothèses, si $u \in M_A$:

$$\begin{cases} \mathbb{P}(u_1 = e) &= 1 & (L.1) \\ \forall t > 1,\ \mathbb{P}(u_t = f \mid u_{t-1} = f) &= 1 & (L.2) \\ \forall t > 1,\ \mathbb{P}(u_t = e) &= 0 & (L.3) \\ \forall t \geqslant n,\ \mathbb{P}(u_t \mid u_{t-1}, \ldots, u_1) &= P(u_t \mid u_{t-1}, \ldots, u_{t-n+1}) & (L.4) \end{cases}$$

Si les n-grammes sont connus, ces hypothèses simplificatrices permettent d'exprimer la probabilité d'un

mot de manière différente :

Propriété L.1.2 : expression de la probabilité
Avec les notations de la définition L.1.1, soit $u \in M_C$, on définit $l(u)$:

$$l(u) = \min\{i \in \mathbb{N} \mid u_i = f\}$$

Par définition de u, $l(u)$ existe et la probabilité de u peut s'exprimer différemment :

$$\mathbb{P}(u) = \begin{cases} \mathbb{P}(u) & \text{si } l(u) < n \\ \mathbb{P}(u_1, \ldots, u_{n-1}) \prod_{t=n}^{l(u)} \mathbb{P}(u_t \mid u_{t-1}, \ldots, u_{t-n+1}) & \text{si } l(u) \geqslant n \end{cases}$$

Démonstration (propriété L.1.2) :
La définition L.1.1 s'inspire de celle d'une chaîne de Markov d'ordre n (voir paragraphe E.6.1, page 286), la démonstration aussi. **(L.1.2)** □

L.2 Estimation

L'estimation des n-grammes s'effectue pour un sous-ensemble $C \subset M_A$ donné, on définit deux types de probabilités :

1. La probabilité de commencer un mot par la séquence x :

$$\forall x \in A^n,\ p_e(x, C) = \widehat{P}\left(u_1^{n-1} = x\right)$$

2. La probabilité de transiter à l'intérieur d'un mot de la séquence x à l'élément y :

$$\forall x \in A^n,\ \forall y \in A,\ \forall t > n,\ p_t(x, y, C) = \widehat{P}\left(u_t = y \mid u_{t-n+1}^{t-1} = x\right)$$

Ces deux probabilités peuvent être estimées à l'aide de l'ensemble C comme suit :

$$p_e(x, C) = \frac{card\left\{u \in C \mid u_1^{n-1} = x\right\}}{card(C)}$$

$$p_t(x, y, C) = \begin{cases} 0 \text{ si } card\left\{(u,t) \mid u \in C,\ n \leqslant t \leqslant l(u),\ u_{t-n+1}^{n-1} = x\right\} = 0 \\ \text{sinon } \dfrac{card\left\{(u,t) \mid u \in C,\ n \leqslant t \leqslant l(u),\ u_{t-n+1}^{n-1} = x,\ u_t = y\right\}}{card\left\{(u,t) \mid u \in C,\ n \leqslant t \leqslant l(u),\ u_{t-n+1}^{n-1} = x\right\}} \end{cases}$$

L.3 Prolongations

L.3.1 Densité d'un dictionnaire

L'idée d'associer une densité à un dictionnaire est tirée de l'article [Govindaraju2002]. Effectuée sur une même base d'images de mots cursifs, la reconnaissance de l'écriture voit ses performances décroître au fur et à mesure que la taille du dictionnaire augmente. Le choix est plus vaste, par conséquent, la possibilité de se tromper est plus grande. Toutefois, la taille n'est pas le seul paramètre à prendre en compte, un dictionnaire dans les mots sont très proches les uns des autres propose de nombreux choix similaires. Soit D un dictionnaire, la densité de $D = (m_1, \ldots, m_N)$ notée $\rho(D)$ est définie par :

$$\rho(D) = \underbrace{\frac{N(N-1)}{\sum_{i \neq j} d_R(m_i, m_j)}}_{v_R(D)} f_R(N) \tag{L.5}$$

$f_R(N)$ est une fonction croissante de N telle que $f_R(N) = (\ln N)^p + \delta_R$ ou $f_R(N) = N^p + \delta_R$ avec $p > 0$. $d_R(m_i, m_j)$ est une distance qui mesure la confusion entre ces deux mots via un système de reconnaissance R, elle sera définie plus loin. L'article [Govindaraju2002] montre de manière pratique que les performances $p_R(D)$ du système de reconnaissance R évoluent linéairement par rapport à la densité [1] :

$$p_R(D) \sim a\rho(D) + b \tag{L.6}$$

La distance $d_R(w_i, w_j)$ est égale à la distance entre les deux modèles de reconnaissance associés aux mots w_i et w_j. Soient deux états e_i^k et e_j^l des modèles de reconnaissances associés aux mots w_i et w_j, ils différent par leurs probabilités d'émission que l'on peut comparer grâce à une distance de Kullback-Leiber. Il est ensuite possible de construire une distance entre graphes de sorte qu'elle soit la somme des distances d'éditions entre tous les chemins du premier graphe et tous ceux du second.

L.3.2 Classes de symboles

Plutôt que de modéliser l'ensemble des n-grammes, il peut paraître judicieux de regrouper certains symboles en classes puis de ne s'intéresser qu'aux transitions entre classes de symboles, ce que proposent les articles [Yamamoto2003] et [Perraud2003]. Jusqu'ici, les n-grammes représentés sont assimilables à des chaînes de Markov, mais les classes de symboles pourraient être les états de la chaîne de Markov cachée. Les mots peuvent par exemple être classés par rapport à leur fonction grammaticale dans la phrase, cette classe serait dans le cas présent l'observation cachée. On peut donc imaginer que les états de la chaîne de Markov représentent des classes de mots et émettent des mots. Le modèle ainsi formé est une modélisation du langage. Soit $D = (m_1, \ldots, m_d)$ une liste de symboles ou plus concrètement de mots, on désire modéliser les séquences de mots. Les n-grammes des paragraphes précédents modélisent la probabilité d'un séquence $S = (s_1, \ldots, s_T)$ par :

$$\mathbb{P}(S) = \mathbb{P}(s_1, \ldots, s_d) \prod_{i=d+1}^{T} \mathbb{P}(s_i \mid s_{i-1}, \ldots, s_{i-d})$$

En classant les mots dans une liste de classes $(C_1, \ldots, C_X)$ considérée comme les états d'une chaîne de Markov cachée, soit $c = (c_1, \ldots, c_T)$ une séquence de classes, la probabilité de la séquence S s'écrit maintenant :

$$\mathbb{P}(S) = \sum_c \left[\prod_{i=1}^{T} \mathbb{P}(s_i \mid c_i)\right] \mathbb{P}(c_1, \ldots, c_d) \left[\prod_{i=d+1}^{T} \mathbb{P}(c_i \mid c_{i-1}, \ldots, c_{i-d})\right]$$

Cette expression est calculable grâce à l'algorithme E.2.3 (page 252).

Alors que l'article [Perraud2003] élabore les classes de manière sémantique (les mots sont classés selon leur fonction grammaticale), l'article [Yamamoto2003] propose une méthode permettant de déterminer le

1. Pour une base donnée, $p_R(D)$ correspond au nombre de mots reconnus correctement sur le nombre de documents dans la base.

nombre de classes ainsi qu'un critère d'évaluation nommé *perplexité* et défini comme suit pour une liste de séquence de symboles $\left(s_1^k, \ldots, s_{T_s}^k\right)_{1 \leqslant k \leqslant K}$:

$$\begin{aligned} H &= -\frac{1}{K} \sum_{k=1}^{K} \ln \mathbb{P}\left(s_1^k, \ldots, s_{T_s}^k\right) \\ P &= 2^H \end{aligned} \tag{L.7}$$

Par rapport à [Perraud2003], l'article [Yamamoto2003] propose une modélisation plus complexe, alliant probabilités de transitions pour les sous-séquences centrales de symboles et probabilité de transitions entre classes pour les sous-séquences au bord. Soit n la dimension des n-grammes et $s = (s_1, \ldots, s_T)$ une séquence de symboles dont les classes associées sont $(c_1, \ldots, c_T)$ (les classes sont connues) :

$$\begin{aligned} \mathbb{P}\left(s_1, \ldots, s_d, s_{d+1}, \ldots, s_{T-d}, s_{T-d+1}, \ldots, s_T\right) = & \prod_{i=1}^{d} \mathbb{P}\left(c_i \mid c_1, \ldots, c_i\right) \ \mathbb{P}\left(s_i \mid s_i\right) \\ & \prod_{i=d+1}^{T-d} \mathbb{P}\left(s_i \mid s_{i-1}, \ldots, s_{i_d}\right) \\ & \prod_{i=T-d+1}^{T} \mathbb{P}\left(c_i \mid c_1, \ldots, c_i\right) \ \mathbb{P}\left(s_i \mid s_i\right) \end{aligned}$$

Dans cette expression, les débuts et fin de mots, supposés moins fiables pour une estimation, sont modélisés par des classes de caractères tandis que pour la partie centrale, les caractères sont directement modélisés.

L.3.3 Choix de la dimension de n-grammes

La définition de la perplexité (L.7) implique nécessaire sa décroissance lorsque la dimension n croît ainsi que le montre la table L.1 regroupant les calculs de perplexité pour différentes valeurs de la dimension. Comme dans toute modélisation, la question du choix de la dimension appropriée se pose.

A l'instar de l'article [Bicego2003], il est possible d'utiliser un critère d'information comme le BIC - ou Bayesian Information Criterion - afin de mettre en rapport la baisse de la perplexité avec le nombre de coefficients ajoutés aux n-grammes lorsqu'on augmente leur dimension. Les notations utilisées sont celles de l'expression (L.7). On définit N_k comme étant le nombre de paramètres libres pour le modèle de dimension k et S représente la somme des longueurs des séquences d'observations. Le meilleur modèle maximise le critère suivant :

$$BIC\left(k\right) = \sum_{k=1}^{K} \ln \mathbb{P}\left(s_1^k, \ldots, s_{T_s}^k\right) - \frac{N_k}{2} \ln S \tag{L.8}$$

La table L.1 montre les résultats obtenus pour un dictionnaire de cinq mille mots anglais courants. Le critère est maximum pour une dimension égale à trois.

Il est possible de raffiner la méthode afin de sélectionner la meilleure dimension locale. Par exemple, dans le cas d'un mot incluant la lettre "Z", il n'est pas nécessaire de connaître la lettre précédant la lettre "Z" pour prévoir celle qui suit. Pour la lettre "Z" les 2-grammes ou bi-grammes suffisent alors qu'avec la lettre

dimension	$\log_2$-perplexité	$\frac{BIC(dimension)}{K}$
2	19,75	−20,29
3	16,20	−19,64
4	12,23	−21,61
5	9,64	−24,12
6	8,81	−26,47
7	8,60	−27,96
8	8,54	−28,69
9	8,52	−28,99
10	8,52	−29,14

Tab. L.1: Log-perplexité estimée pour différentes dimensions et sur un dictionnaire de 5000 mots anglais employé de manière courante et contenant en moyenne entre cinq et six lettres. La perplexité décroît lorsque la dimension augmente tandis que le critère BIC préconise une dimension égale à trois pour laquelle il est minimum.

"A", il est préférable de choisir des 3-grammes ou tri-grammes. Il s'agit donc ici d'estimer un modèle de n-grammes avec un n assez grand puis de supprimer certains coefficients jusqu'à ce que le critère BIC ait atteint son minimum. Dans ce cas, les n-grammes peuvent être considérés comme les états d'une chaîne de Markov, être classés par ordre décroissant de probabilité a posteriori [2] puis être supprimés selon cet ordre tant que le critère BIC croît.

L.3.4 Groupe de lettres récurrents

Lors du traitement des erreurs de segmentation [3], la reconnaissance de l'écriture nécessite la sélection des groupes de lettres les plus fréquents. Si le "." signifie le début ou la fin d'un mot ou l'espace, ".a.", ".de.", "tion." reviennent fréquemment dans la langue française. On s'intéresse ici à des probabilités de transition entre des groupes de plusieurs lettres. Jusqu'à présent, les modèles de n-grammes présentés estime la probabilité de la lettre suivant sachant la ou les lettres précédentes. Dans ce cas, on cherche la probabilité des lettres suivantes sachant un passé d'une ou plusieurs lettres.

La table L.2 présente un extrait des noms utilisés dans la bande dessinée *Astérix le Gaulois* dans laquelle les suffixes *ix.* et *us.* sont couramment employés. Il est naturel d'envisager la probabilité de ces triplets - deux lettres plus la fin du mot - sachant la lettre précédente. Il reste à estimer des probabilités comme $\mathbb{P}(u \mid t)$ et $\mathbb{P}(us. \mid t)$.

Pour ce faire, on définit un alphabet étendu $A = (s_1, \ldots, s_N)$ incluant tous les groupes de lettres dont on veut estimer les transitions. On définit également $s + t$ comme étant la concaténation de deux sympboles de l'alphabet, par exemple : $t + us = tus$. Pour un mot donné, on désigne par $E_A(m)$ toutes les manières possibles d'écrire le mot m en utilisant les symboles inclus dans A. On dispose d'une base de mots $(m_1, \ldots, m_K)$. Les probabilités de transitions sont alors définies par :

2. Annexes : voir paragraphe F.1.3, page 297
3. Annexes : voir paragraphe 4.7, page 124

Astérix	Obélix	Panoramix	Abraracourcix
Assurancetourix	Agécanonix	Tragicomix	Cétautomatix
Idéfix	Plaintecontrix	Ordralfabétix	Pneumatix
Plantaquatix	Elèvedelix	Analgésix	Monosyllabix
Uniprix	Linguistix	Arrierboutix	Obélodalix
Harenbaltix	Choucroutgarnix	Bellodalix	Zérozérosix
Allégorix	Boulimix	Porquépix	Aplusbégalix
Théorix	Homéopatix	Tournedix	Squinotix
Cumulonimbus	Pleindastus	Fleurdelotus	Langélus
Yenapus	Roideprus	Fanfrelus	Faipalgugus
Détritus	Diplodocus	Garovirus	Cubitus
Diplodocus	Infarctus	Suelburnus	Saugrenus
Volfgangamadéus	Soutienmordicus	Épinedecactus	Cétinconsensus

Tab. L.2: Prénoms gaulois et romains extraits de la bande dessinée *Astérix le Gaulois*. Pour cet extrait, $\mathbb{P}(ix. \mid t) = \frac{3}{10}$, $\mathbb{P}(us. \mid t) = \frac{2}{10}$.

$$N = \sum_{k=1}^{K} card\left(E_A\left(m_k\right)\right)$$

$$\mathbb{P}(s) = \frac{1}{N} \sum_{k=1}^{K} \left[\sum_{(s_1,\ldots,s_n)\in E_A(m_k)} \mathbb{1}_{\{s_1=s\}} \right] \tag{L.9}$$

$$\mathbb{P}(t \mid s) = \frac{\sum_{k=1}^{K} \left[\sum_{(s_1,\ldots,s_n)\in E_A(m_k)} \sum_{i=2}^{n} \mathbb{1}_{\{s_{i-1}=s \text{ et } s_i=t\}} \right]}{\sum_{k=1}^{K} \left[\sum_{(s_1,\ldots,s_n)\in E_A(m_k)} \sum_{i=2}^{n} \mathbb{1}_{\{s_{i-1}=s\}} \right]} \tag{L.10}$$

Cet ensemble n'est pas forcément réduit à un seul élément (voir table L.3). Avec ce formalisme, il est maintenant possible d'exprimer la probabilité d'un mot m comme étant :

$$\mathbb{P}(m) = \sum_{(s_1,\ldots,s_n)\in E_A(m)} \mathbb{P}(s_1) \prod_{i=2}^{n} \mathbb{P}(s_i \mid s_{i-1}) \tag{L.11}$$

alphabet	B - BO - O - OA - A
mot	BOA
écriture 1	B - OA
écriture 2	BO - A
écriture 3	B - O - A

Tab. L.3: Différentes manières d'écrire le mot "BOA".

Cet outil permet d'estimer des probabilités de transitions entre des modèles de Markoc cachées modélisant des groupes de lettres [4] présentés au paragraphes 4.7 (page 124). L'ensemble $E_A(m)$ n'est ici pas précisé

4. Annexes : voir paragraphe G.3, page 332

et est supposé être l'ensemble des écritures possibles et admises par l'aphabet A. Cependant, pour la reconnaissance de l'écriture, toutes les écritures ne sont pas équiprobables puisqu'une écriture est définie comme étant le résultat de la segmentation en graphèmes dont les erreurs (voir figure 3.2, page 41) déterminent l'ensemble $E_A(m)$.

L.3.5 Lissage des n-grammes

L'article [Béchet2004] propose une méthode permettant de lisser des probabilités de transitions entre mots et d'obtenir par exemple des probabilités non nulles de transitions entre couples de caractères non représentés dans l'ensemble d'estimation. A partir des nombres de transitions c_{ij} d'un contexte h_i (un ou plusieurs mots) vers un mot w_j, les auteurs construisent un espace vectoriel E_C de représentation des contextes. L'objectif est de construire des compteurs de transitions augmentés a_{ij} prenant en compte non seulement les transitions du contexte h_i vers le mot w_j mais aussi les transitions des contextes proches de h_i vers un mot proche de w_j, la proximité étant mesurée dans l'espace E_C par une distance. L'article montre empiriquement que les performances dans une tâche de reconnaissance sont d'autant plus accrues par un tel lissage que la taille du vocabulaire est grande.

Annexe M

Receiving Operator Characteristic (ROC)

M.1 Définition

La courbe ROC est utilisée pour déterminer les performances d'un système de classification, de reconnaissance (voir figure M.1).

Définition M.1.1 : ROC
On suppose que chaque expérience donne lieu à deux informations : $(X, \theta) \in \mathbb{R} \times \{0, 1\}$. X est ce qu'on appelle un score, θ détermine si l'expérience est réussie (valeur 1) ou a échoué (valeur 0). Pour n expériences, on dispose de N couples $((X_1, \theta_1), \ldots, (X_n, \theta_n))$. On définit en fonction d'un seuil $s \in \mathbb{R}$:

1. Le taux de reconnaissance : $R(s) = \frac{1}{n} \sum_{i=1}^{n} \theta_i \, \mathbb{1}_{\{X_i \geqslant s\}}$
2. Le taux d'erreur : $E(s) = \frac{1}{n} \sum_{i=1}^{n} (1 - \theta_i) \, \mathbb{1}_{\{X_i \geqslant s\}}$

Le courbe ROC est le graphe $(E(s), R(s))$ lorsque s varie dans $\mathbb{R}$.

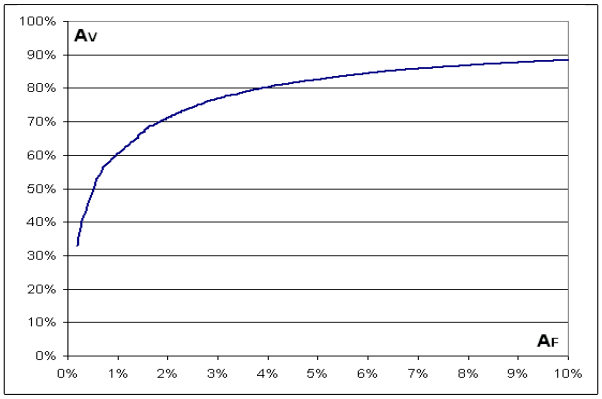


Fig. M.1: Exemple de courbe ROC : seule la partie correspondant aux faibles taux d'erreur est intéressante. Dans cet exemple, pour 1% de documents reconnus par erreur, 70% le sont correctement.

Remarque M.1.2: fonctions monotones
On remarque que les fonctions $s \longrightarrow E(s)$ et $s \longrightarrow R(s)$ sont dÕcroissantes toutes deux. Elles sont donc

inversibles.

Remarque M.1.3: score aléatoire
Dans le cas où la variable aléatoire θ est indépendante de la variable X, la courbe ROC est une droite reliant les points $(0,0)$ et $(1-p,p)$ où $p = \mathbb{P}(\theta = 1)$. Ceci signifie que la connaissance du score X n'apporte pas d'information quant à la réussite de l'expérience.

On utilise aussi une autre courbe presque équivalente à la première. Celle-ci ne met plus en rapport le nombre d'erreurs avec le nombre de documents dont le score dépasse un certain seuil mais le nombre d'erreurs avec le nombre de documents traités ou *taux de lecture*.

Définition M.1.4 : ROC'
Les notations sont identiques à celles de la définition M.1.1. On définit en fonction d'un seuil $s \in \mathbb{R}$:

1. Le taux de lecture : $R'(s) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \geqslant s\}}$
2. Le taux d'substitution : $E'(s) = \frac{1}{R'(s)} \sum_{i=1}^{n} (1-\theta_i)\, \mathbb{1}_{\{X_i \geqslant s\}}$

La courbe ROC' est le graphe $(E'(s), R'(s))$ lorsque s varie dans $\mathbb{R}$.

Le paragraphe M.4.2 explique pour cette seconde courbe, quoique plus couramment utilisée, se prête moins bien à la méthode décrite au paragraphe M.3 et permettant d'obtenir des intervalles de confiance. En effet, contrairement à la courbe ROC définie en M.1.1, les suites (E') et (R') ne sont pas monotones (voir paragraphe M.3).

M.2 Aire sous la courbe

On suppose pour cette partie que les scores sont tous continus. Dans le cas d'un score discret, la courbe ROC est alors une fonction continue par morceaux.

M.2.1 Estimateur

L'aire sous la courbe est un critère pour évaluer la pertinence d'un score. Plus l'aire est grande, plus le score X est pertinent. Il est donc important de mesurer cette aire, aussi appelée en anglais *area under the ROC curve* (AUC). Mais avant de donner un estimateur de cette courbe, il est préférable de donner une seconde définition de la courbe ROC (équivalente à la définition M.1.1).

Définition M.2.1 : ROC (2)
On suppose que Y est la variable aléatoire des scores des expériences qui ont réussi. X est celle des scores des expériences qui ont échoué. On suppose également que tous les scores sont indépendants. On note F_X et F_Y les fonctions de répartition de ces variables. On définit en fonction d'un seuil $s \in \mathbb{R}$:

1. Le taux de reconnaissance : $R(s) = 1 - F_Y(s)$
2. Le taux d'erreur : $E(s) = 1 - F_X(s)$

La courbe ROC est le graphe $(E(s), R(s))$ lorsque s varie dans $\mathbb{R}$.

On peut alors en déduire le théorème suivant :

Théorème M.2.2 : aire sous la courbe
On utilise les notations de la définition M.2.1. L'aire sous la courbe ROC est égale à $\mathbb{P}(Y > X)$.

Démonstration (théorème M.2.2) :
On note f_X la densité de la variable X et f_Y celle de la variable Y. On peut alors définir la probabilité $\mathbb{P}(Y > X)$ par une intégrale :

$$P(Y > X) = \int_x \int_y f_X(x)\, f_Y(y)\, \mathbb{1}_{\{y>x\}} dxdy \tag{M.1}$$

On note F_X la fonction de répartition de X [1]. On pose comme changement de variable : $u = F_X(x)$. On en déduit que $du = f_X(x)dx$. La variable aléatoire $U = F_X(X)$ est uniforme et comprise dans $[0,1]$ [2].

$$\begin{aligned} P(Y > X) &= \int_x f_X(x)dx \int_y f_Y(y)\, \mathbb{1}_{\{y>x\}} dy \\ &= \int_u du \int_y f_Y(y)\, \mathbb{1}_{\left\{y>F_X^{-1}(u)\right\}} dy \\ &= \int_u du\, \mathbb{P}\left(Y > F_X^{-1}(u)\right) \end{aligned}$$

Or si $u = F_X(s) = E(s)$, alors $F_X^{-1}(u) = s$ et $\mathbb{P}\left(Y > F_X^{-1}(u)\right) = R'(s)$. Par conséquent :

$$P(Y > X) = \int_u du\, \mathbb{P}\left(Y > F_X^{-1}(u)\right) = \int_u du\, R'(F_X^{-1}(u)) \tag{M.2}$$

Cette dernière expression est l'aire recherchée.

(M.2.2) □

Ce théorème nous permet de définir un estimateur pour l'aire sous la courbe ROC à l'aide des U-statistiques

1. $F_X(x) = \int_{-\infty}^{x} f_X(u)du$
2. Soit X une variable aléatoire de densité f et de fonction de répartition F. Si $U = F(X)$, alors :

$$\mathbb{P}(U \leqslant t) = \mathbb{P}(F(X) \leqslant t) = \mathbb{P}\left(X \leqslant F^{-1}(t)\right) = F\left(F^{-1}(t)\right) = t$$

La variable U est de loi uniforme sur $[0,1]$. De plus, soit g une fonction intégrable quelconque, on pose $u = F(x)$ et :

$$\int_{\mathbb{R}} g(x)\, f(x)\, dx = \int_{[0,1]} g(F^{-1}(u))\, du$$

de Mann-Whitney (voir [Saporta1990]).

Corollaire M.2.3 : estimateur de l'aire sous la courbe ROC'
On dispose des scores $(Y_1, \ldots, Y_n)$ des expériences qui ont réussi et $(X_1, \ldots, X_m)$ les scores des expériences qui ont échoué. On suppose également que tous les scores sont indépendants. les scores (Y_i) sont identiquement distribués, il en est de même pour les scores (X_i). Un estimateur de l'aire A sous la courbe ROC' est :

$$\hat{A} = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{\{Y_j > X_i\}} + \frac{1}{2} \mathbb{1}_{\{Y_j = X_i\}} \tag{M.3}$$

Démonstration (corollaire M.2.3) :
La démonstration est évidente : $\mathbb{E}\left(\hat{A}\right) = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{P}\left(Y_j > X_i\right) + \frac{1}{2}\mathbb{P}\left(X = Y\right) = \mathbb{P}\left(Y > X\right) + \frac{1}{2}\mathbb{P}\left(Y = X\right)$.
Dans le cas où X ou Y sont continues, $\mathbb{P}\left(X = Y\right) = 0$. **(M.2.3)** □

M.2.2 Intervalles de confiance

Il est possible de déterminer un intervalle de confiance pour cet estimateur. Le théorème central limite nous permet de dire que cet estimateur tend vers une loi normale lorsque n et m tendent vers l'infini.

Corollaire M.2.4 : variance de l'estimateur
On note $P_X = \mathbb{P}\left(X < \min\left\{Y_i, Y_j\right\}\right)$ et $P_Y = \mathbb{P}\left(\max\left\{X_i, X_j\right\} < Y\right)$. X_i et X_j sont de même loi que X, Y_i, Y_j sont de même loi que Y. La variance de l'estimateur $\hat{A}$ défini par (M.3) est :

$$\mathbb{V}\left(\hat{A}\right) = \frac{\hat{A}(1-\hat{A})}{nm} \left[1 + (n-1)\frac{P_Y - \hat{A}^2}{\hat{A}(1-\hat{A})} + (m-1)\frac{P_X - \hat{A}^2}{\hat{A}(1-\hat{A})}\right] \tag{M.4}$$

Démonstration (corollaire M.2.4) :
Cette démonstration n'est vrai que dans le cas continu. Par conséquent, $\mathbb{P}\left(X = Y\right) = 0$. On calcule tout d'abord $\mathbb{E}\left(\hat{A}^2\right)$ et on utilise le fait que $\mathbb{V}\left(\hat{A}\right) = \mathbb{E}\left(\hat{A}^2\right) - \hat{A}^2$.

$$\begin{aligned}
\hat{A}^2 &= \frac{1}{n^2\, m^2} \left[\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{\{X_i < Y_j\}}\right]^2 = \frac{1}{n^2\, m^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l=1}^{n} \mathbb{1}_{\{X_i < Y_j\}} \mathbb{1}_{\{X_k < Y_l\}} \\
&= \frac{1}{n^2\, m^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{\{X_i < Y_j\}} + \frac{1}{n^2\, m^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k \neq i} \mathbb{1}_{\{X_i < Y_j\}} \mathbb{1}_{\{X_k < Y_j\}} + \\
&\quad \frac{1}{n^2\, m^2}\, \frac{1}{n^2\, m^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{l \neq j} \mathbb{1}_{\{X_i < Y_j\}} \mathbb{1}_{\{X_i < Y_l\}} + \frac{1}{n^2\, m^2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k \neq i} \sum_{l \neq j} \mathbb{1}_{\{X_i < Y_j\}} \mathbb{1}_{\{X_k < Y_l\}}
\end{aligned} \tag{M.5}$$

On en déduit que :

$$\begin{aligned}
\mathbb{E}\left(\hat{A}^2\right) &= \frac{\hat{A}}{nm} + \frac{n-1}{nm}\,\mathbb{P}\left(\max\{X_i, X_k\} < Y_j\right) + \\
&\quad \frac{m-1}{nm}\,\mathbb{P}\left(X_i < \min\{Y_j, Y_l\}\right) + \frac{nm-n-m-1}{nm}\,\hat{A}^2 \\
\mathbb{V}\left(\hat{A}^2\right) &= \frac{1}{nm}\left[\hat{A} + (n-1)P_Y + (m-1)P_X - (n+m+1)\hat{A}^2\right] \\
&= \frac{1}{nm}\left[\hat{A} + (n-1)\left(P_Y - \hat{A}^2\right) + (m-1)\left(P_X - \hat{A}^2\right) + \hat{A}^2\right]
\end{aligned}$$

On retrouve l'expression cherchée.

(M.2.4) □

M.3 Intervalles de confiance pour la courbe

Les systèmes de reconnaissance sont souvent ajustés de telle manière que le taux d'erreur soit constant, par exemple 1%. C'est la proportion de documents reconnus qui détermine la performance de ce système. L'objectif ce paragraphe est de déterminer un intervalle de confiance du taux de reconnaissance pour un taux d'erreur fixé, ce que la figure M.1 ne peut pas retourner.

M.3.1 Construction de la courbe ROC

Ce premier paragraphe détaille la manière dont est construite une courbe ROC (voir définition M.1.1).

Algorithme M.3.1 : courbe ROC
On suppose qu'on dispose d'un ensemble de points $(X_i, \theta_i) \in \mathbb{R} \times \{0,1\}$ pour $i \in \{1, \ldots, n\}$. X_i est le score obtenu pour l'expérience i, θ_i vaut 1 si elle a réussi et 0 si elle a échoué. On suppose également que cette liste est triée par ordre croissant : $\forall i,\ X_i \leqslant X_{i+1}$. On souhaite également tracer k points sur la courbe, on détermine pour cela k seuils $\{s_1, \ldots, s_k\}$ définis par : $\forall j, s_k = X_{\frac{j\,k}{n}}$.
On construit ensuite les points (R_j, E_j) définis par :

$$R_j = \frac{1}{n}\sum_{i=1}^{n} \theta_i \mathbb{1}\!\!1_{\{X_i \geqslant s_j\}} \text{ et } E_j = \frac{1}{n}\sum_{i=1}^{n} (1-\theta_i)\ \mathbb{1}\!\!1_{\{X_i \geqslant s_j\}} \tag{M.6}$$

La courbe ROC est composée de l'ensemble $R_{OC} = \{(E_j, R_j)\ |\ 1 \leqslant j \leqslant k\}$.

Les deux suites $(R_j)_j$ et $(E_j)_j$ sont toutes les deux croissantes d'après leur définition. Une autre de ces courbe est données par la figure M.2. A partir de cet ensemble de points, le taux de lecture correspondant

à 1% d'erreur est alors calculé à l'aide d'une interpolation, linéaire par exemple.

Algorithme M.3.2 : taux de reconnaissance
On cherche un taux de reconnaissance pour un taux d'erreur donné. On dispose pour cela d'une courbe ROC obtenue par l'algorithme M.3.1 et définie par les points $R_{OC} = \{(e_j, r_j) \mid 1 \leqslant j \leqslant k\}$. On suppose ici que $(e_1, r_1) = (1, 1)$ et $(e_k, r_k) = (0,)$. Si ce n'est pas le cas, on ajoute ces valeurs à l'ensemble R_{OC}.
Pour un taux d'erreur donné e^*, on cherche j^* tel que :

$$e_{j^*} \leqslant e^* \leqslant e_{j^*+1} \tag{M.7}$$

Le taux de reconnaissance ρ cherché est donné par :

$$\rho = \frac{e^* - x_{j^*}}{x_{j^*+1} - x_{j^*}} \left[r_{j^*+1} - r_{j^*}\right] + r_{j^*} \tag{M.8}$$

Il ne reste plus qu'à détailler la méthode *bootstrap*.

M.3.2 Méthode boostrap

Une seule courbe ROC ne permet d'obtenir qu'un seul taux. On cherche ici à construire plusieurs courbes ROC à partir de la même expérience de façon à obtenir plusieurs taux de reconnaissance pour le même taux d'erreur. De cette manière, il sera possible de déterminer un intervalle de confiance. On s'inspire pour cela des méthodes de *bootstrap*.

Algorithme M.3.3 : courbe ROC, méthode boostrap
On dispose toujours du nuage de points $E = (X_i, \theta_i) \in \mathbb{R} \times \{0, 1\}$ avec $i \in \{1, \ldots, n\}$. On choisit $C \in \mathbb{N}$ le nombre de courbes ROC qu'on désire tracer. Pour chaque courbe $c \in \{1, \ldots, C\}$:

1. On construit un nouvel ensemble $(X'_i, \theta'_i)_{1 \leqslant i \leqslant n}$ construit par un tirage aléatoire dans l'ensemble E avec remise.
2. L'algorithme M.3.1 permet de constuire la courbe R^k_{OC}.
3. L'algorithme M.3.2 permet ensuite de déterminer un taux de reconnaissance ρ_k pour le taux d'erreur e^*.

La liste $(\rho_1, \ldots, \rho_C)$ est triée par ordre croissant. Les quantiles sont ensuite utilisés pour déterminer l'intervalle de confiance $[\rho_1, \rho_2]$ du taux de reconnaissance pour le taux d'erreur e^* de telle sorte que :

$$\mathbb{P}\left(\rho \in [\rho_1, \rho_2]\right) = 1 - \alpha \tag{M.9}$$

On prend généralement $\alpha = 5\%$.

La figure M.2 illustre les résultats obtenus par l'algorithme M.3.3.

M.3.3 Aire sous la courbe

La méthode bootstrap peut elle aussi être appliquée pour calculer un intervalle de confiance pour l'aire sous la courbe (AUC).

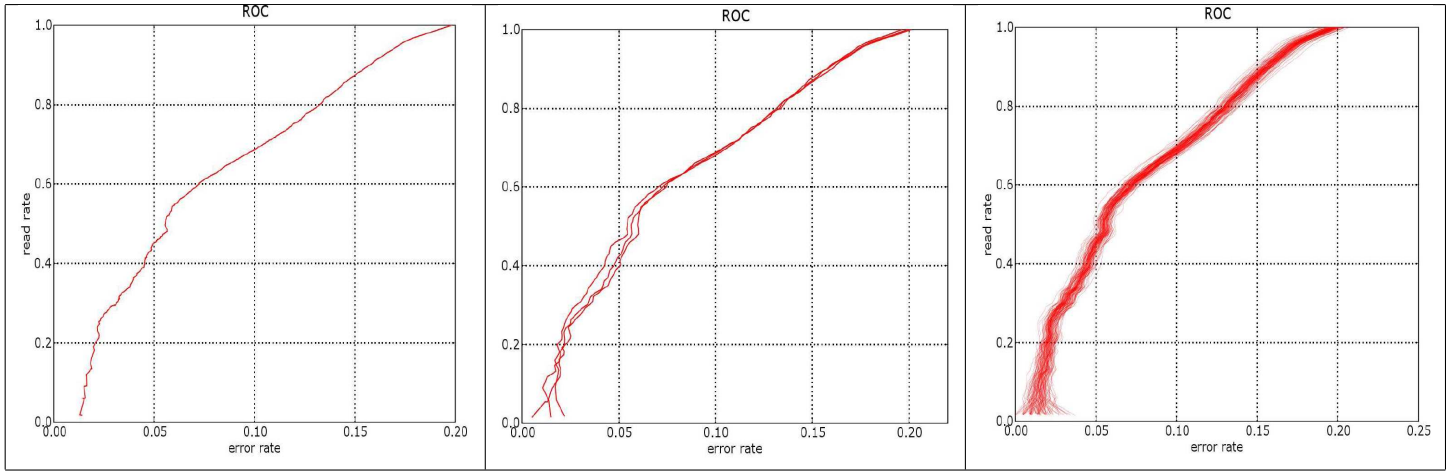


Fig. M.2: La première image est celle d'une courbe ROC, la seconde représente toutes celles obtenues par la méthode bootstrap pour trois courbes. La troisième image superpose cent courbes. Moins il y a de points pour estimer une partie de la courbe, plus les courbes sont espacées. Ces courbes ont été construites avec 12000 points. Le taux de lecture pour 1% d'erreur est égal à 68,09%. L'intervalle de confiance à 95% est $[\,66,10\%,\ 70,16\%\,]$ (construit avec 500 courbes). Moyenne (68,25) et médiane (68,12) sont sensiblement égales au taux calculé sur la première courbe construite sans tirage aléatoire. L'écart-type est $1,10$, cela donne un intervalle de confiance équivalent au précédent si on considère que la moyenne des taux suit asymptotiquement une loi normale. Cette expérience a été reproduite plusieurs fois et ces bornes sont assez stables contrairement ($\pm 0,05\%$) aux extremas ($\pm 1\%$). Ces courbes sont les courbes ROC' (définition M.1.4) et non ROC (définition M.1.1).

M.4 Pour aller plus loin

M.4.1 Distribution des scores mauvais et bons

On appelle un mauvais score un score associé à un mauvais résultat, de même, un bon score est le score d'un bon résultat. Si le score est une probabilité, on s'attend à trouver les bons scores regroupés autour de la valeur 1. Si le score est un mauvais score, il devrait être plus proche de zéro. La figure M.4 montre des distributions obtenus pour deux problèmes différents. Dans les deux cas, le but recherché est la détermination d'un seuil séparant le score d'un bon résultat de celui d'un mauvais résultat. Lorsque ceci n'est pas possible, le score ne peut correspondre à un quelconque critère confiance.

M.4.2 Taux de lecture ou de reconnaissance

La plupart des courbes montrées en exemple font référence à une courbe mettant en relation un taux de lecture et un taux de substitution (définition M.1.4) car elles sont plus facilement interprétable. Par exemple, pour un taux de substitution de 1%, si on a 70% en taux de lecture, cela signifie que sur 100 documents, le système va en accepter 70 et parmi ces 70, 1% seront mal traités. Le taux de substitution est un taux d'erreur rapporté à un taux de lecture donné. L'inconvénient du taux de lecture rapporté au taux de substitution est que la méthode développée au paragraphe M.3 ne s'applique plus aussi bien car pour un taux de substitution donné, il peut exister plusieurs taux de lecture, ce que montre la figure M.5. Cette même figure montre les répercussions sur le calcul des intervalles de confiance.

On peut démontrer que la courbe taux de lecture / taux de substitution n'est pas une courbe ni monotone ni inversible. Pour cela on dispose d'une suite de couple (X_i, θ_i) croissante selon les X_i. θ_i vaut 1 si l'expérience a réussi, 0 sinon. Pour un seuil donné s, on note $E'(s)$ le taux de substitution et $R'(s)$ le taux

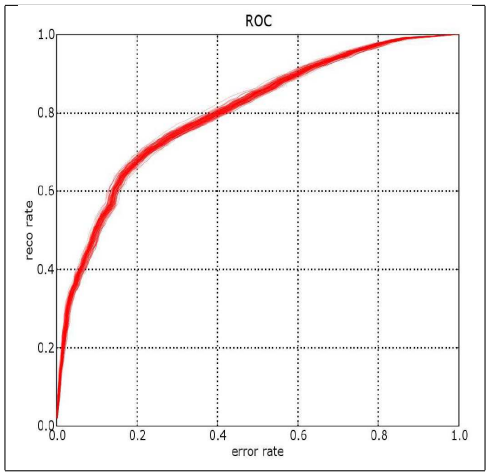


Fig. M.3: Courbe ROC (définition M.1.1) obtenue pour 100 tirages aléatoires. L'aire sous la courbe est égale à 0.80 et l'intervalle de confiance à 95% mesurée par la méthode bootsrap est : [0.79, 0.80]. Les extremas sont presque identiques à ces chiffres.

de lecture, on obtient :

$$R'(s) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{X_i \geqslant s\}}$$
$$E'(s) = \frac{1}{n\,R'(s)}\sum_{i=1}^{n} (1-\theta_i)\ \mathbb{1}_{\{X_i \geqslant s\}}$$

On écrit différemment ces expressions en supposant que $X_{i(s_1)-1} < s_1 \leqslant X_{i(s_1)}$:

$$R'(s_1) = \frac{n - i(s_1)}{n}$$
$$E'(s_1) = \frac{1}{n - i(s_1)} \sum_{i=i(s_1)}^{n} (1-\theta_i)$$

On suppose maintenant que $X_{i(s_2)-1} < s_2 \leqslant X_{i(s_2)}$ et $i(s_1)+1 = i(s_2)$:

$$R'(s_2) = \frac{n - i(s_2)}{n} < R'(s_1)$$
$$E'(s_2) = \frac{1}{n - i(s_2)} \sum_{i=i(s_2)}^{n} (1-\theta_i) = \frac{1}{n - i(s_2)} \frac{n - i(s_1)}{n - i(s_1)} \left(-\left(1-\theta_{i(s_1)}\right) + \sum_{i=i(s_1)}^{n} (1-\theta_i) \right)$$
$$= -\frac{\left(1-\theta_{i(s_1)}\right)}{n - i(s_2)} + \frac{\sum_{i=i(s_1)}^{n} (1-\theta_i)}{n - i(s_1)} \frac{n - i(s_1)}{n - i(s_2)} = -\frac{\left(1-\theta_{i(s_1)}\right)}{n - i(s_2)} + E'(s_1) \frac{n - i(s_1)}{n - i(s_2)}$$

Si on suppose que $\theta_{i(s_1)} = 1$, autrement dit, l'expérience s_1 a réussi, on en déduit que :

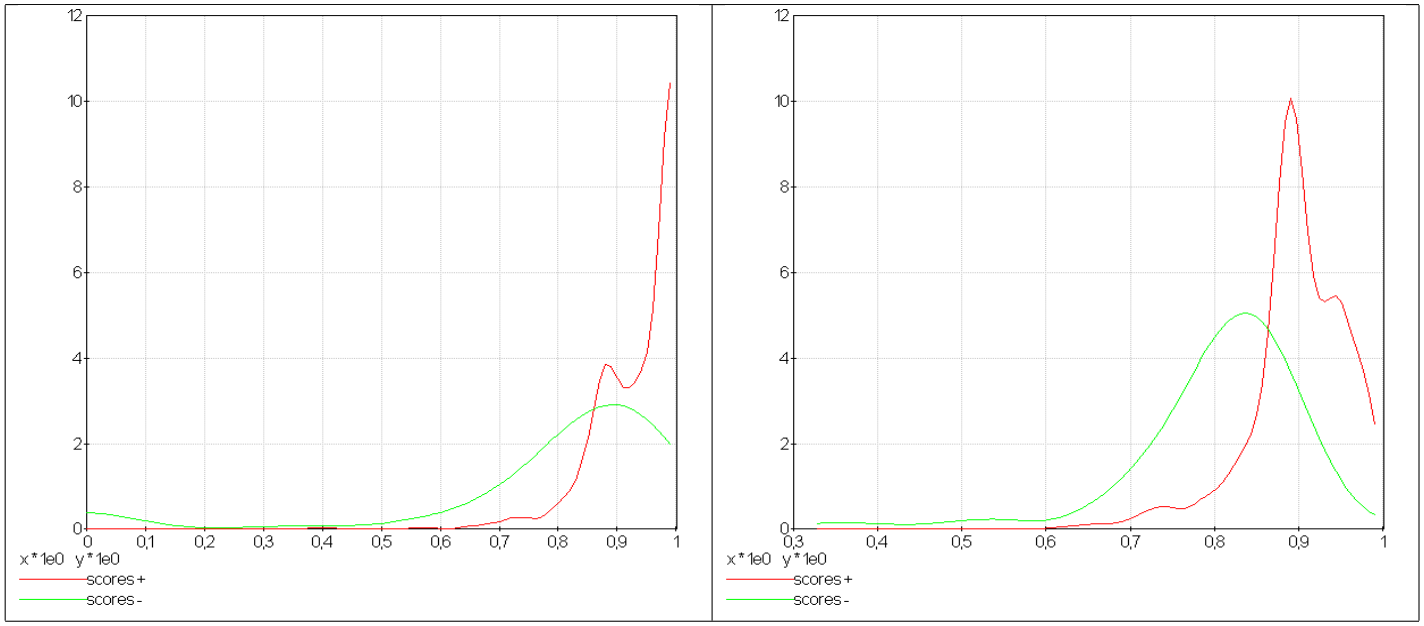


Fig. M.4: Distribution des bons et mauvais scores. La première courbe montre deux distributions qui se chevauchent même si les bons scores semblent plus concentrés autour des grandes valeurs. Le seconde courbe montre un problème mieux séparable. L'existence d'un seuil entre un bon et un mauvais score est plus plausible.

$$E'(s_2) = E'(s_1)\frac{n - i(s_1)}{n - i(s_2)} = E'(s_1)\frac{n - i(s_2) + 1}{n - i(s_2)} > E'(s_1)$$

En revanche si $\theta_i = 0$:

$$E'(s_2) = E'(s_1)\left(1 + \frac{1}{n - i(s_2)}\right) - \frac{1}{n - i(s_2)} = E'(s_1) + \frac{E(s_1) - 1}{n - i(s_2)} < E'(s_1)$$

Il n'existe donc pas toujours une fonction f reliant $R'(s)$ à $E'(s)$ à moins de construire cette courbe de telle sorte qu'elle soit monotone en ne choisissant qu'une sous-suite $(E'(X_i), R'(X_i))_i$ qui vérifie cette hypothèse.

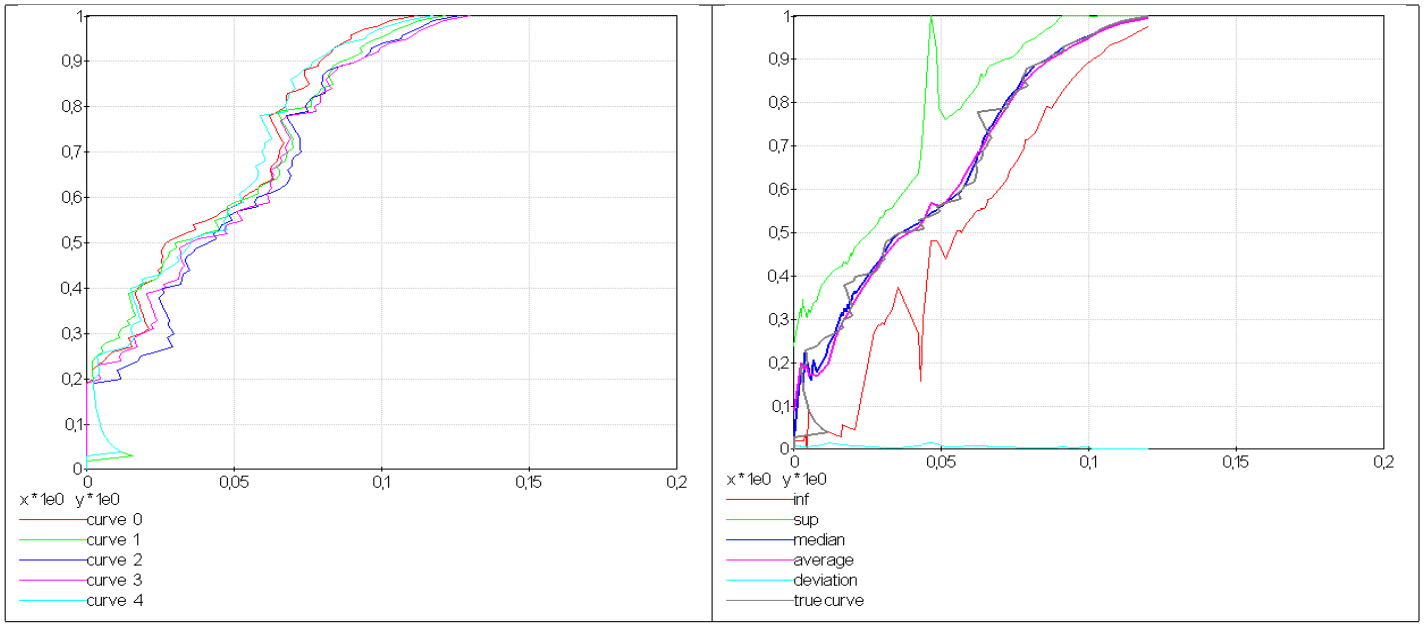
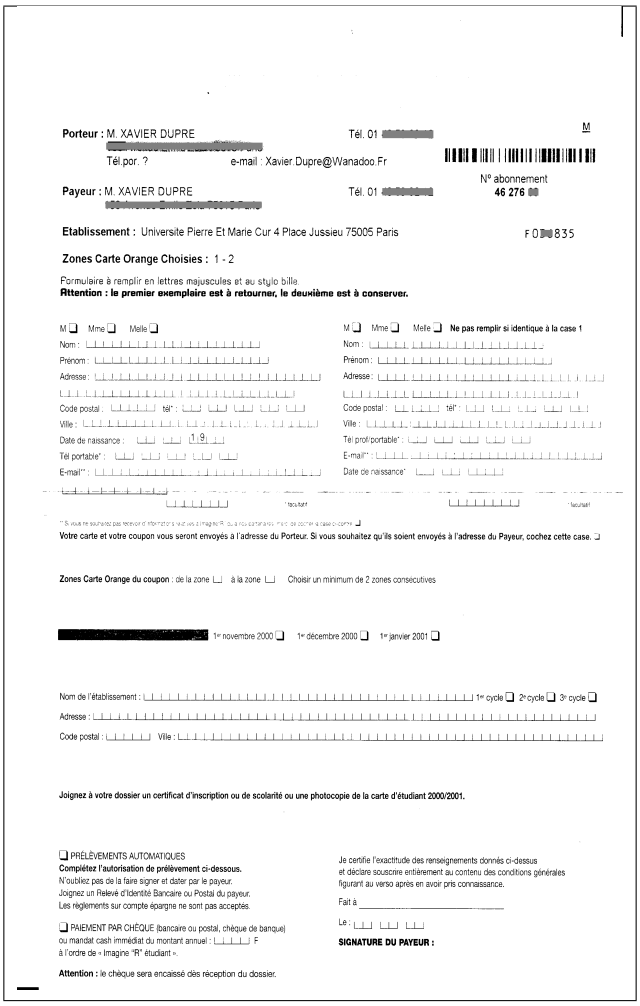


Fig. M.5: La première image montre 5 courbes taux de lecture / taux de substitutions. Les courbes ne sont pas monotones et montre qu'il existe parfois plusieurs taux de lecture pour un même taux de substitution. Comme le calcul des intervalles de confiance fait intervenir une interpolation linéaire, lorsque les courbes sont trop cahotiques, le calcul retourne des valeurs fausses.

Annexe N

Analyse de documents



Porteur : M. XAVIER DUPRE Tél. 01 [redacted]

Tél.por. ? e-mail : Xavier.Dupre@Wanadoo.Fr

Payeur : M. XAVIER DUPRE Tél. 01 [redacted]

N° abonnement 46 276 [redacted]

Etablissement : Universite Pierre Et Marie Cur 4 Place Jussieu 75005 Paris F 0[redacted]835

Zones Carte Orange Choisies : 1 - 2

Formulaire à remplir en lettres majuscules et au stylo bille.
Attention : le premier exemplaire est à retourner, le deuxième est à conserver.

M ❑ Mme ❑ Melle ❑
Nom :
Prénom :
Adresse :
Code postal : tél* :
Ville :
Date de naissance : (19)
Tél portable* :
E-mail** :

M ❑ Mme ❑ Melle ❑ **Ne pas remplir si identique à la case 1**
Nom :
Prénom :
Adresse :
Code postal : tél* :
Ville :
Tel prof portable* :
E-mail** :
Date de naissance*

* facultatif

** Si vous ne souhaitez pas recevoir d'informations sur les produits Imagine "R" ou nos partenaires, merci de cocher la case ci-contre ❑

Votre carte et votre coupon vous seront envoyés à l'adresse du Porteur. Si vous souhaitez qu'ils soient envoyés à l'adresse du Payeur, cochez cette case. ❑

Zones Carte Orange du coupon : de la zone ☐ à la zone ☐ Choisir un minimum de 2 zones consecutives

[redacted] 1er novembre 2000 ❑ 1er décembre 2000 ❑ 1er janvier 2001 ❑

Nom de l'établissement : 1er cycle ❑ 2e cycle ❑ 3e cycle ❑
Adresse :
Code postal : Ville :

Joignez à votre dossier un certificat d'inscription ou de scolarité ou une photocopie de la carte d'étudiant 2000/2001.

❑ PRÉLÈVEMENTS AUTOMATIQUES
Complétez l'autorisation de prélèvement ci-dessous.
N'oubliez pas de la faire signer et dater par le payeur.
Joignez un Relevé d'Identité Bancaire ou Postal du payeur.
Les règlements sur compte épargne ne sont pas acceptés.

❑ PAIEMENT PAR CHÈQUE (bancaire ou postal, chèque de banque)
ou mandat cash immédiat du montant annuel : F
à l'ordre de « Imagine "R" étudiant ».

Attention : le chèque sera encaissé dès réception du dossier.

Je certifie l'exactitude des renseignements donnés ci-dessus
et déclare souscrire entièrement au contenu des conditions générales
figurant au verso après en avoir pris connaissance.

Fait à

Le :

SIGNATURE DU PAYEUR :

Fig. N.1: Exemple de document imprimé, un formulaire à l'intérieur duquel le texte forme l'essentiel de l'information contenue, c'est le texte qu'on souhaite localiser puis reconnaître de façon à extraire un numéro d'abonnement, de téléphone, une adresse, ...

L'analyse de documents consiste à localiser l'information textuelle à partir d'une image de document, le plus souvent au format A4. L'objectif visé dans cette partie est de localiser les mots ou les lignes de mots, afin de résumer l'image d'une page à l'ensemble des images de mots qu'elle contient. La figure N.1 en est un exemple.

Ce problème est abordé par l'article [Liu2000] qui expose une méthode de classification basée sur l'extraction de formulaire, ceux-ci étant définis par l'ensemble des traits des tableaux qui le composent. La classification s'apparente à un appariement de structures. Cet article utilise une méthode de détection des traits developpée dans l'article [Yu1996]. Le paragraphe N.1 aborde ces articles.

N.1 Bibliographie

N.1.1 Méthodes à seuils

L'article [Yu1996] propose une méthode qui permet de détecter les lignes sur un document, de les nettoyer puis de reconstruire les caractères chevauchant ces lignes, qu'ils soient imprimés ou manuscrit. Le principe repose sur un *Block Adjacency Graph*. Les éléments de ce graphe sont les ensembles contigüs de pixels appartenant à une même ligne (voir figure N.2). Chaque bloc est relié aux autres autres blocs de la ligne supérieure ou inférieure ayant plusieurs pixels voisins. Les lignes ont une structure particulière à l'intérieur de ce graphe qui est aussi utilisé pour reconstruire les caractères tronqués une fois que les blocs appartenant à des lignes ont été retirés.

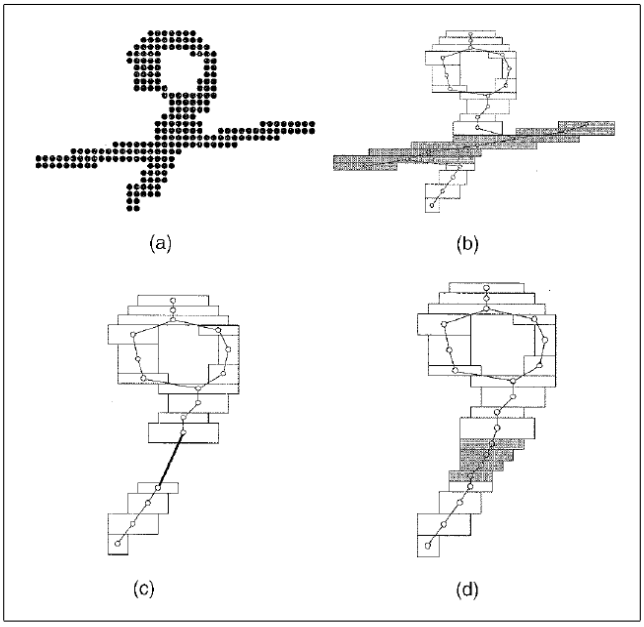


Fig. N.2: Figure extraite de [Yu1996], un *Block Adjacency Graph* est construit afin de détecter les lignes puis à reconstruire le caractère une fois la ligne nettoyée.

L'article [Liu2000] repart des résultats obtenus dans [Yu1996] concernant la détection des lignes. La mméthode cherche à reconstruire le schéma des éléments linéaires d'une page (voir figure N.3). Ce schéma est ensuite comparé à une liste de schémas existants afin de l'identifier. C'est un système de classification basé sur les éléments linéaires d'un document. La distance entre deux schémas linéaires développée dans cet article offre une certaine souplesse puisqu'elle prend en compte les traits manquants, les traits non détectées, ...

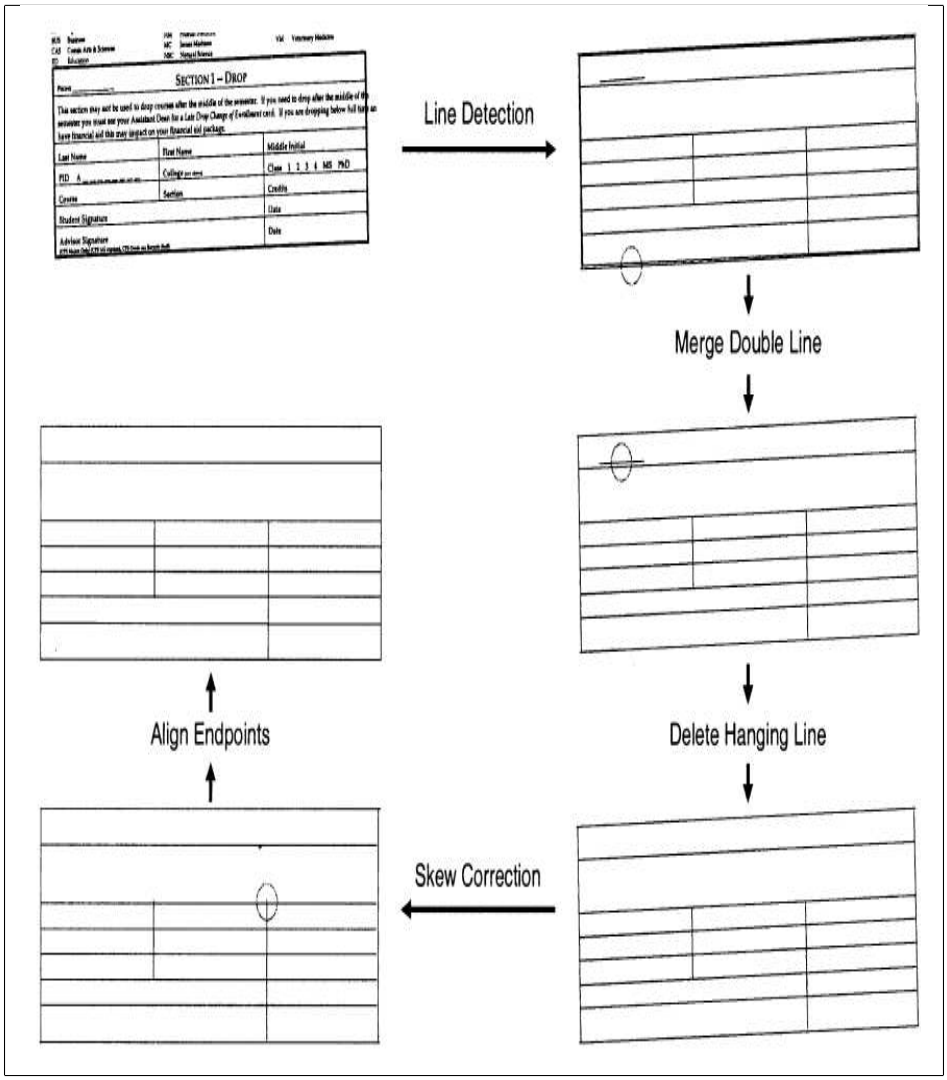


Fig. N.3: Figure extraite de [Liu2000], les lignes obtenues dans [Yu1996] sont prolongées afin que leurs d'extrémités appartiennent à une ligne détectée. Les traits parallèles sont tantôt supprimés, tantôt fusionnés. L'objectif visé est d'obtenir une sorte de schéma des segments contenus dans une page.

En apparté, l'article [Zheng2001] propose une méthode intéressante de détection de l'inclinaison d'un document à partir du moment ou celui-ci contient des lignes (elle est également expliquée dans [Zheng2001]).

$$R_i\left(x_i, y_{s_i}, y_{e_i}\right) = \left\{ (x,y) \;\middle|\; \begin{array}{l} p(x,y) = 1,\ \forall (x,y) \in \{x_i\} \times [y_{s_i}, y_{e_i}] \\ \text{et } p(x_i, y_{s_i} - 1) = p(x_i, y_{e_i} + 1) = 0 \end{array} \right. \tag{N.1}$$

Un pixel $p(x,y)$ est noir s'il contient la valeur 1 et 0 s'il est blanc. La propriété $R_i\left(x_i, y_{s_i}, y_{e_i}\right)$ détermine si une succession de pixels noirs sur la même ligne fait partie d'un segment de ligne telle qu'il serait s'il avait été tracé par un algorithme tel que celui de Bresenham. Un tel segment est appelé *directly signle-connected chain* (DSCC). Poour chaque de ces segments, on définit :

$$\overline{x} = \frac{1}{\sum\limits_{x,y} p(x,y)} \sum_{x,y} x p(x,y) \tag{N.2}$$

$$\overline{y} = \frac{1}{\sum\limits_{x,y} p(x,y)} \sum_{x,y} y p(x,y) \tag{N.3}$$

$$\mu_{mn} = \sum_{x,y} (x - \overline{x})^m (y - \overline{y})^n p(x,y) \tag{N.4}$$

$$d_\mu = \mu_{20} - \mu_{02} \tag{N.5}$$

$$\theta = \frac{1}{2} \tan^{-1} \left(\frac{2\mu_{11}}{d_\mu} \right) \tag{N.6}$$

$$a = \sqrt{\frac{2\left[\mu_{20} + \mu_{02} + \sqrt{d_\mu^2 + 4\mu_{11}^2}\right]}{\mu_{00}}} \tag{N.7}$$

$$b = \sqrt{\frac{2\left[\mu_{20} + \mu_{02} - \sqrt{d_\mu^2 + 4\mu_{11}^2}\right]}{\mu_{00}}} \tag{N.8}$$

On ne garde que les DSCC qui vérifient deux conditions. Les seuils T_1 et T_2 sont définis à partir d'expériences.

$$\max(a,b) < T_1 \tag{N.9}$$

$$\frac{a}{b} > T_2 \tag{N.10}$$

$$\theta \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \tag{N.11}$$

L'angle le plus représenté au sein des DSCC est supposé être l'orientation du document.

N.1.2 Méthodes probabilistes

La thèse [Zheng2006] aborde des méthodes probabilistes pour résoudre les problèmes présentés lors du paragraphe précédent. Elles sont plus coûteuses à mettre en œuvre puisqu'elles nécessitent l'estimation de modèles probabilistes tels que les modèles Markov cachés. La première méthode proposée est celle de la détection des lignes sur un document. On suppose que son orientation a été corrigée, puis le document a été projeté horizontalement pour avoir un histogramme représentant le nombre de pixels par ligne. C'est à partir de cette séquence que seront déterminées la position des lignes horizontales à l'aide d'un modèle de Markov caché à deux états qui corresondent à l'état ligne et non-ligne. Le nettoyage des lignes est ensuite assuré par des méthodes de traitements d'image à seuil. L'avantage et l'inconvénient de cette méthode est son apprentissage. Si le flux de documents à traiter est homogène (peu de types différents de documents), l'apprentissage est une bonne approche. Pour un flux de documents hétérogènes (nombreux types différents de documents), cela semble plus discutable. Les expériences présentées dans la thèse [Zheng2006] ne permettent d'apprécier la robustesse de la méthode face à un flux hétérogène.

N.2 Segmentation d'une page de texte imprimé

N.2.1 Hypothèses

On part du principe qu'une page, aussi bruitée soit-elle, va contenir un grand nombre de composantes connexes qui seront des caractères imprimés. Cette démarche ne marchera pas pour une page contenant trop de manuscrit ou un bruit se rapprochant plus d'une texture que d'un nuage uniforme de composantes connexes composées de quelques pixels. On suppose également que les différentes polices utilisées ne sont pas de tailles trop dissemblables. En résumé, on suppose que toute ligne de texte imprimée contient suffisamment de caractères formant seul une composante connexe.

N.2.2 Principe général

On commence par déterminer la densité des tailles $s = (s_x, s_y)$ des composantes connexes. On aboutit fréquemment à une densité illustrée par la figure N.4. Soit C l'ensemble des composantes connexes, on note $s(c) = (s_x(c), s_y(c)) \in \mathbb{N}^2$ la taille de la composante connexe c. La densité est plus un histogramme 2D puisqu'on se place dans un espace discret :

$$\forall s \in \mathbb{N}^2,\ h(s) = \#\left\{s(c) = s \mid c \in C\right\} \tag{N.12}$$

On note $s^* = \left(s_x^*, s_y^*\right) = \arg\max h(c)$. Dans la majeure partie des cas, ce pic s^* devrait correspondre à la lettre la plus fréquente dans la police la plus fréquente. Pour schématiser, cette lettre a toutes les chances d'être un "e" minuscule dans le cas d'un document imprimé en langue française. A partir de ce pic, on en déduit un intervalle probable contenant les tailles des autres polices de caractères présents dans cette même page. Cet intervalle est fixé à $[5, 6s_x^*] \times \left[5, 3s_y^*\right]$.

On classe ensuite les composantes connexes en trois ensembles, les petites C^-, les composantes acceptables C^* et les grosses composantes connexes notées C^+ définies comme suit :

$$C^- = \{c \in C \text{ tel que } s(c) = (s_x, s_y) \text{ vérifie } \min(s_x, s_y) \leqslant 5\} \tag{N.13}$$
$$C^* = \left\{c \in C \text{ tel que } s(c) = (s_x, s_y) \text{ vérifie } 5 \leqslant s_x \leqslant 6s_x^* \text{ et } 5 \leqslant s_y \leqslant 3s_y^*\right\} \tag{N.14}$$
$$C^+ = \left\{c \in C \text{ tel que } c \notin C^- \cup C^*\right\} \tag{N.15}$$

On suppose que l'ensemble C^* suffit à déterminer la structure du document. Cela signifie que les polices acceptées pour former cette structure ont au moins cinq pixels de côtés et sont au pire six fois plus larges et trois fois plus hautes que la police la plus représentée. La section N.2.5 montre quelques cas qui ne suivront pas localement ces hypothèses sans pour autant les remettre en cause au niveau de la page. La détermination de la structure de la page va entièrement reposer sur l'ensemble C^* ou presque, on procède en quatre étapes :

1. Détermination des ensembles C^-, C^*, C^*.
2. Détection des traits, tableaux, dans l'ensemble C^+ puis récupération des composantes connexes connectées à ces traits (voir section N.4).
3. L'ensemble C^* reçoit les composantes connexes détachées des traits dont les dimensions satisfont la condition (N.14).
4. Construction d'un graphe à partir de C^* qui permettra d'exrtraire les lignes (voir paragraphe N.2.3).

Fig. N.4: L'intensité noire d'un pixel signifie une taille fortement représentée dont les dimensions sont données par les coordonnées du pixels : (x, y) signifie une largeur x et une hauteur y. Le point de coordonnées $(1, 1)$ est en haut à gauche. La densité de la taille des caractères dépend du nombre de polices différentes présentes sur l'image. Une police est caractérisée par la présence d'une ou deux lignes horizontales très proches (hauteur des minuscules puis des majuscules). Si la hauteur varie peu pour une même police, la largeur est plus dispersée, le "m" étant souvent plus large que le "i".

N.2.3 Construction d'un graphe

A partir de l'ensemble de composantes connexes C^*, on construit un graphe ne reliant que les plus proches voisins selon les quatre directions, droite, haut, gauche, bas. La figure N.5 montre la position des voisins acceptables pour une composante connexe. On note $G(c) = (g_x(c), g_y(c))$ le centre de gravité de la composante connexe c définie par le centre de la boîte englobante $B(c) = (x_1(x), y_1(c), x_2(c), y_2(c))$. Ce plus, ces voisins ne doivent pas être trop éloignés les uns des autres. On rappelle que $\left(s_x^*, s_y^*\right)$ est la taille de la police la plus représentée. Pour c, on définit dans ces quatre directions les voisins les plus proches :

$$V^*_{left}(c) = \arg\inf \{g_x(d) - x_2(c) \mid d \in C,\ (g_x(d) - x_2(c), g_y(d)) \in [0, 6s_x^*] \times [y_1(c), y_2(c)]\} \quad \text{(N.16)}$$

$$V^*_{right}(c) = \arg\inf \{x_1(c) - g_x(d) \mid d \in C,\ (x_1(c) - g_x(d), g_y(d)) \in [0, 6s_x^*] \times [y_1(c), y_2(c)]\} \quad \text{(N.17)}$$

$$V^*_{top}(c) = \arg\inf \left\{y_1(c) - g_y(d) \mid d \in D,\ (g_x(d), y_1(d) - g_y(d)) \in [x_1(c), x_2(c)] \times \left[0, 3s_y^*\right]\right\} \quad \text{(N.18)}$$

$$V^*_{bottom}(c) = \arg\inf \left\{g_y(d) - y_2(c) \mid d \in D,\ (g_x(d), g_y(d) - y_2(d)) \in [x_1(c), x_2(c)] \times \left[0, 3s_y^*\right]\right\} \quad \text{(N.19)}$$

Ces quatre valeurs peuvent être égales à $\emptyset$ s'il n'existe pas de composantes connexes acceptables dans l'une des quatre directions. Le résultat est illustré par la figure N.6.

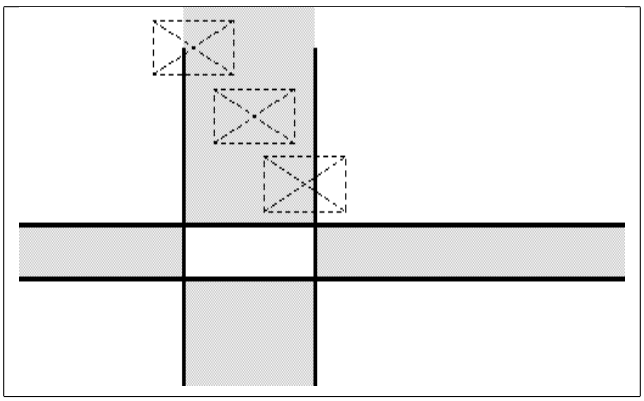


Fig. N.5: Une zone a pour voisin l'ensemble des autres composantes connexes dont le centre de gravité est situé dans les zones grises. Quatre zones grises pour quatre directions privilégiées par le sens de lecture, haut, bas, gauche, droite.

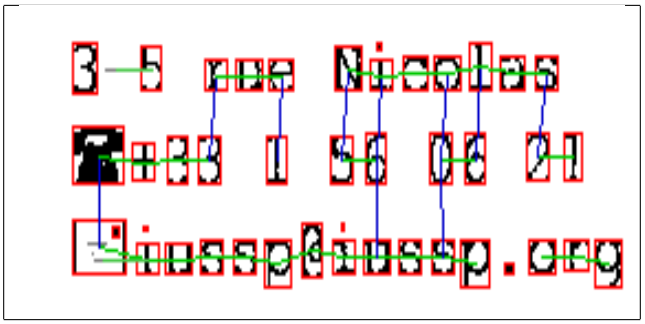


Fig. N.6: Les traits entre composantes connexes représentent les liens droite et bas d'une composante à l'autre. Tous ne sont pas présents, les voisins ne doivent pas être trop éloignés les uns des autres.

N.2.4 Regroupement et construction des lignes

Les voisins V^*_{top} et V^*_{bottom} ont pour objectif la construction des paragraphes mais avant, la construction des lignes n'utilise que les liens V^*_{left} et V^*_{right}. Le graphe seulement composé de ces liens suffit à former les lignes qui sont en fait les composantes connexes de ce graphe. Autrement dit, pour construire une ligne, on prend n'importe quelle composante connexe et on parcourt la ligne vers la droite en passant d'un voisin V^*_{right} au suivant. A ce stade, on suppose que les lignes ainsi formées sont dans leur ensemble cohérentes avec le document. Elles sont seulement incomplètes ou parfois en plusieurs morceaux mais elles ne sont pas fausses. Ce travail se poursuit par différentes étapes :

1. incorporation des petites coposantes connexes (ensemble C^-)
2. regroupement des lignes
3. traitement des grosses composantes connexes (ensemble C^+)
4. nettoyage des lignes

La liste de ces traitements n'est pas exhaustive, elle peut dépendre des application utilisant l'analyse de documents. L'essentiel est de conserver une structure récurrente partagée par tous les algorithmes, comme ce graphe par exemple.

N.2.4.1 Cas des petites composantes connexes (ensemble C^-)

L'étape suivante est de recoller les petites composantes connexes (ensemble C^-) à ces lignes si elles ne sont pas trop éloignées de celles-ci. Dans ces petites composantes connexes, on trouve parfois des lettres "i" ou "l" ou "1", des signes de ponctuation ".", ",", " :", ... Elles sont incorporées aux lignes selon les mêmes conditions que celles qui définissent les voisins dans la section N.2.3.

N.2.4.2 Lignes en plusieurs morceaux

La section N.2.5 donne quelques exemples de lignes découpées. Les cas courants sont une séparation quelques pixels trop grandes comparées aux seuils imposés dans la section N.2.3. On peut néanmoins calculer quelques statistiques concernant les lignes sur lesquelles s'appuieront les méthodes de fusion des lignes.

On définit une ligne par une séquence de composantes connexes triées de gauche à droite : $L = (c_1, \ldots, c_n)$. On définit l'écart entre deux composantes connexes par : $\delta_i = max\,(0, x_1(c_{i+1}) - x_2(c_i))$. On construit la suite : $\delta(L) = (\delta_i, \ldots, \delta_{n-1})$. On note également h_i la hauteur de la composante c_i.

$$\widehat{s(L)} = mediane\left(\delta_i, \ldots, \delta_{n-1}\right) \tag{N.20}$$

$$\widehat{w(L)} = \mathbb{E}\left(\delta_i \mathbb{1}_{\left\{\delta_i \geqslant \widehat{s(L)}\right\}}\right) \tag{N.21}$$

$$\widehat{h(L)} = \mathbb{E}\left(h_i\right) \tag{N.22}$$

$$\widehat{\sigma_h(L)} = \sqrt{\mathbb{E}\left(h_i - \widehat{h(L)}\right)^2} \tag{N.23}$$

$\widehat{s(L)}$ est la médiane des écarts, on peut s'attendre à ce qu'elle soit proche de l'espace inter-caractère. $\widehat{w(L)}$ est la moyenne des écarts au-dessus de la médiane, on peut également s'attendre à ce qu'elle s'approche de l'écart entre mots dans la même ligne. $\widehat{h(L)}$ est la moyenne des hauteurs, elle doit correspondre à une hauteur moyenne de la ligne. On pourrait également s'intéresser à l'épaisseur moyenne du trait sur cette ligne.

Fusionner deux morceaux de lignes, l'un à gauche, l'autre à droite, c'est vérifier si certains critères comme ceux exposés ci-dessus sont homogènes d'un morceaux à l'autre. Ceci devrait également permettre de distinguer l'imprimé du cursif. Un classifieur devrait pouvoir faire ce travail à condition de lui dire quoi fusionner.

Il arrive simplement aussi que les boîtes englobantes de deux lignes se superposent, que l'une soit incluse dans l'autre. Ce genre n'est pas toujours évident car une des composantes connexes peut être un morceaux de tableau et suivrent simplement les bords de la boîte qui le contient.

Par exemple, certains grosses composantes connexes sont trop larges pour faire partie de l'ensemble C^*. Toutefois, leur hauteur est sembable aux composantes connexes à gauche ou à droite. Pour décider si cette grosse composante connexe de hauteur h doit être ajoutée à la ligne L, on vérifie que $h \in \left[\widehat{h(L)} - 1.96\widehat{\sigma_h(L)},\ \widehat{h(L)} + 1.96\widehat{\sigma_h(L)}\right]$ pour la ligne la plus proche de h. Cela signifie qu'on fait l'hypothèse que la hauteur d'une composante connexe à l'intérieur d'une ligne suit une loi normale.

N.2.4.3 Traitement des grosses composantes connexes (ensemble C^+)

Le cas des grosses composantes connexes regroupent quatre sous-cas :

1. les titres : cas de plusieurs grosses composantes connexes écrites dans une police démesurément grande
2. les intitulés : la plupart des lettres d'une ligne ont été sélectionnées excepté une ou deux trop grande de quelques pixels
3. les lettres connectées en largeur : ces composantes sont grandes en largeur mais aussi hautes que les autres
4. les lettres connectées en hauteur : il s'agit d'une composante connexe partagée par deux lignes, il n'y a pas d'autres moyens que de la découper

Il est facile pour les trois premiers cas de prévoir quelque chose qui va marcher efficacement dans 90% des cas. Ce sont les 10% restants du dernier cas qui posent problème, difficile à prévoir et qui dépendent des projets. La partie N.2.5 donnent quelques exemples de ce type.

N.2.4.4 Traits inclus dans l'ensemble C^*

L'ensemble C^* même si on suppose qu'il contient dans la grande majorité des lettres, certains éléments de tableaux, coupés du reste par quelques pixels manquants, arrivent à se glisser dans cet ensemble et fausse

la détection des lignes. Il faut détecter que ce sont des traits à moins que le reconnaisseur de caractères sache distinguer la lettre "l" d'une barre verticale de tableau, plus fine.

N.2.4.5 Nettoyage des lignes

Parmi les composantes connexes de l'ensemble C^*, certains contiennent des traits même si leur taille leur a permis d'être intégrées dans l'ensemble C^*. Là encore, on suppose que ces mauvaises composantes sont en nombre suffisamment petit pour ne pas trop perturber la détection des lignes. On peut néanmoins constuire certains indicateurs permettant de détecter ce genre de composantes. $L = (c_1, \ldots, c_n)$ est une ligne. On note :

$$\widehat{y_1(L)} = mediane\,\{y_1(c_1), ..., y_1(c_n)\} \tag{N.24}$$

$$\widehat{y_2(L)} = mediane\,\{y_2(c_1), ..., y_2(c_n)\} \tag{N.25}$$

Si la ligne contient suffisamment de composantes connexes, $\left|\widehat{y_1(L)} - \widehat{y_2(L)}\right|$ est un estimateur de la taille des caractères de cette ligne. Toute composante connexe dépassant largement cette estimation est susceptible d'être bruitée ou de contenir des traits par exemple. De même toute composante hors de l'intervalle $\left[\widehat{y_1(L)}, \widehat{y_2(L)}\right]$ est probablement aussi du bruit. D'une manière générale, les médianes sont plus consistantes que les moyennes car les lignes contiennent entre 5 et 20 caractères, une composante aberrante modifie de façon non négligeable n'importe quelle moyenne.

N.2.5 Composantes à segmenter

Cette section est avant tout une série d'exemples pour lesquels les caractères imprimées se superposent à de l'écriture manuscrite, une signature, des traits... voir figures N.7, N.8, N.9, N.10, N.11, N.12, N.13, N.14, N.15, N.16, N.17.

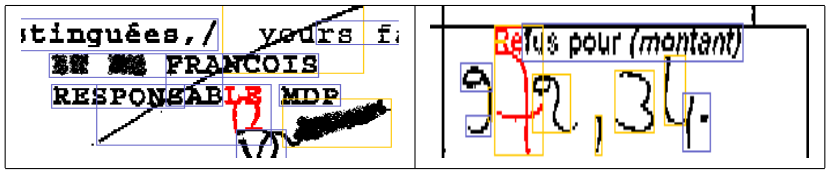


Fig. N.7: La détection des lignes est correcte mais celle-ci doit être prolongée pour casser les composantes connexes qui incluent à la fois un caractère imprimé et un caractère manuscrit.

Fig. N.8: Vidéo inverse : il faut calculer la densité et si elle est trop élevée, inverser l'image.

Fig. N.9: Les traits sont bien détectés excepté un qui est inclus dans la zone de texte. Pas de solution immédiate hormis la hauteur du trait plus élevée que n'importe quelle lettre.

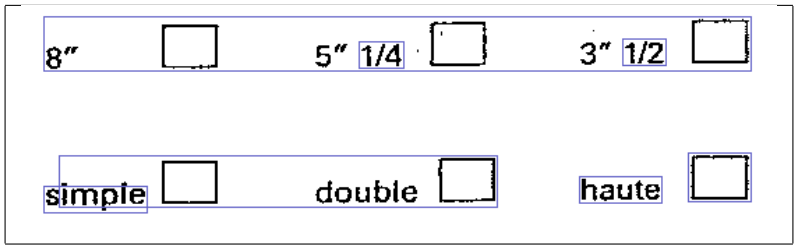


Fig. N.10: Les cases à cocher ne sont pas bien détectées, il faudra peut-être inclure un traitement spécifique de cette forme.

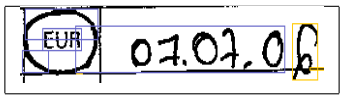


Fig. N.11: Ecriture cursive imperméable à la régularité nécessaire au bon fonctionnement de l'algorithme.

N.3 Segmentation en mots

On peut supposer qu'une même composante connexe n'intercepte pas deux mots différents. Il est probable que ce cas ne se produise pas souvent, est-il négligable ou tolérable ? On peut supposer également qu'une ligne est écrite avec la même police, mais ceci peut-être testé statistiquement. Il est même envisageable de segmenter en polices distinctes à condition qu'il n'y en ait pas cinquante différentes sur une même ligne à l'aide par exemple de critères tels que ceux définis au paragraphe N.2.4.2.

Aujourd'hui la segmentation en mots repose sur un réseau de neurones qui apprend si oui ou non il existe une séparation inter-mots entre deux composantes connexes (ou non). Il devrait être possible de construire un système identique, il suffit d'annoter.

N.4 Détection des tableaux

N.4.1 Idées

L'idée de base est de comparer le squelette vectorisé de l'image aux segments détectés par la méthode exposée par Jérémie Jakubowicz (*http ://www.xavierdupre.fr/exposes/index.html*) : seuls les segments communs sont conservés. Deux segments sont considérés comme communs si : (voir figure N.18)

1. ils sont parallèles
2. la distance qui les séparent (perpendiculaires) n'est pas plus grande que $k = 3$ fois l'épaisseur du trait
3. leur longueur commune est supérieure à n fois l'épaisseur de trait ($n = 15$)

N.4.2 Squelette et vectorisation

Le squelette utilisé ici est un squelette en 8-connexité (algorithme B.4.1, page 165) obtenu par érosion. Etant donné que le résultat de cette squelettisation a peu d'importance sur des parties d'un document autres que celles contenant du texte, il est possible d'améliorer la qualité de ce squelette en lissant préalablement les contours des caractères en utilisant la méthode décrite au paragraphe 3.4.2 page 51. Comme ce squelette est celui d'une zone de texte, il est possible de prolonger ces nettoyages en tenant compte de

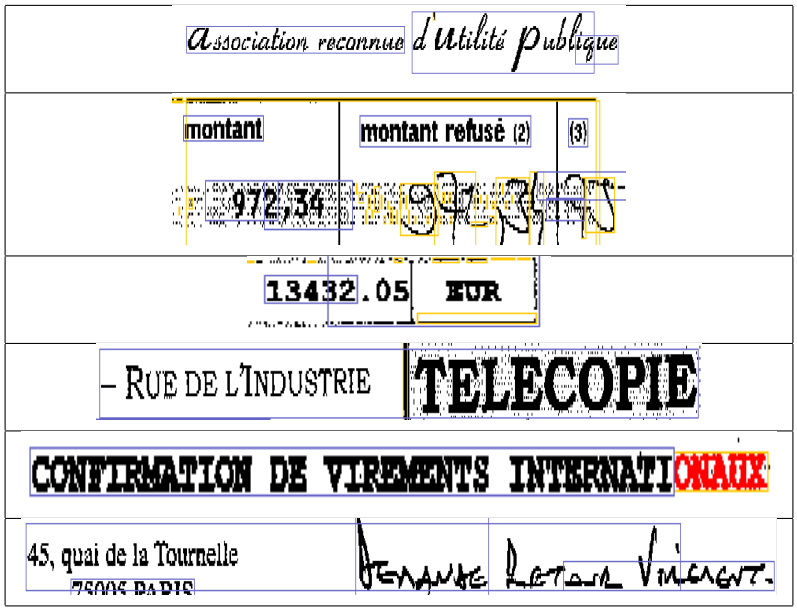


Fig. N.12: Les erreurs simples à corriger : les lignes ne sont pas associées, d'autres conditions de fusion devraient permettre d'inclure ces cas.

Fig. N.13: Sans traitement spécifique, ce bruit gêne la formation d'une ligne car les pixels se touchent et forment une seule composante connexe. Une solution envisageable serait de compter le nombre de transitions pixels noir/blanc pour dissocier la zone de texte de la zone de bruit.

l'épaisseur du trait. On obtient l'algorithme de squelettisation suivant :

Algorithme N.4.1 : vectorisation de texte
Cet algorithme n'est qu'une séquence d'autres décrits dans d'autres chapitres.

1. La première est une étape de lissage décrite au paragraphe 3.4.2 page 51.
2. La seconde est la squelettisation proprement dite, une squelettisation discrète par érosion décrite par l'algorithme B.4.1, page 165.
3. La troisième étape est l'estimation de l'épaisseur du trait. Etant donné que le texte est censé présenter une épaisseur homogène sur toute sa longueur, l'estimation peut être faite à l'aide d'une carte de distance, celle de l'algorithme B.3.4, page 164. L'épaisseur du trait est alors la moyenne des valeurs obtenues pour cette carte pour l'ensemble ds pixels appartenant au squelette obtenu.
4. L'étape suivante est une étape de nettoyage. On définit les extrémités du squelette par l'ensemble des pixels seulement connectés à un seul autre pixel du squelette. Ces extrémités sont rabottées sur une longueur égale à l'épaisseur du trait sans que la connexité du squelette ne soit brisée.
5. La dernière étape est celle de la vectorisation. Il utilise l'algorithme N.4.2 qui suit.

L'algorithme de vectorisation peut également tenir compte du fait que l'image à squelettiser est une image

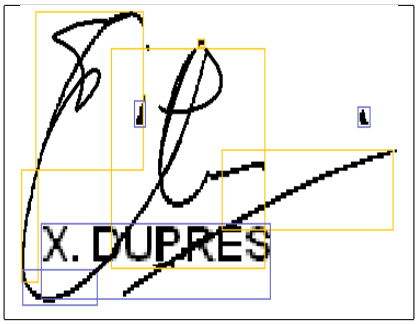


Fig. N.14: Cursif sur imprimé, les composantes connexes sont brouillées par la présence d'un trait cursif. Le prolongement des lignes devraient permettre de s'en tirer.

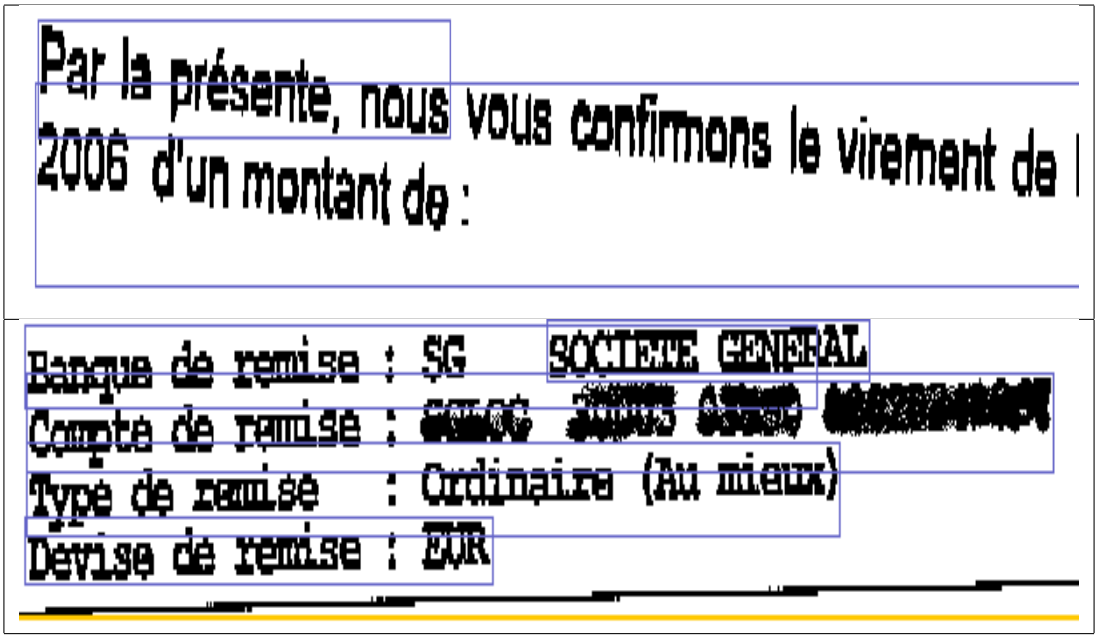


Fig. N.15: L'algorithme supporte une inclinaison légère mais pas trop quand même comme sur le second exemple.

de texte.

Algorithme N.4.2 : vectorisation de squelette
On utilise d'abord l'algorithme B.7.2, page 180 avec un seuil égal à un multiple de l'épaisseur du trait (aux alentours de 2). Malgré tout, il peut arriver que cela ne suffise pas. Lors de la squelettisation de tableaux, les angles droits doivent être bien marqués. On utilise alors une autre condition pour couper un arc de squelette en deux vecteurs : une condition angulaire. Un arc de squelette est représenté par une séquence de pixels : $(p_1, \ldots, p_n)$. Pour un indice i donné, on calcule l'angle $\widehat{p_{i-2e}p_ip_{i+2e}}$ où e est l'épaisseur du trait. Si cette angle n'est pas dans l'intervalle $\left[-\frac{3\pi}{3}, -\frac{5\pi}{3}\right]$, alors c'est un angle remarquable et qui doit marquer la frontière entre deux arcs linéaires.

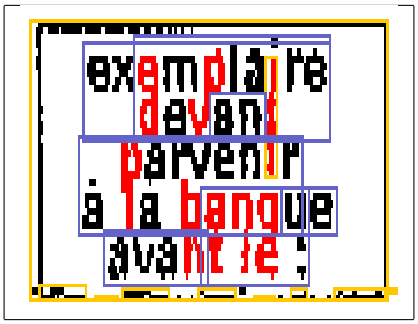


Fig. N.16: Deux lignes à segmenter mais trop de composantes connexes se touchent. Peut-être qu'un filtre sur le rapport probable hauteur/largeur permettrait de détecter ces exemples.

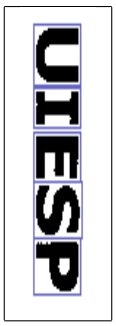


Fig. N.17: Ligne verticale, pas prise en compte pour le moment mais il est envisageable de s'y intéresser.

N.4.3 Détection des segments

Afin d'être rapide, cette détection suppose que les seuls traits à détecter sont des traits horizontaux et verticaux provenant de tableaux. L'image à analyser doit donc être redressée si besoin est. Le paragraphe 3.4.1, page 49, donne une idée de la marche à suivre. L'algorithme qui suit ne présente que la détection de traits

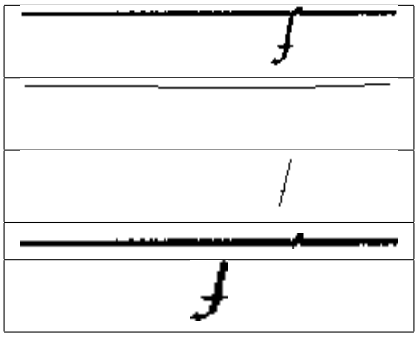


Fig. N.18: Première image : celle à découper. Seconde et troisième images : segmentation des morceaux du squelette en trait et non traits. Quatrième et cinquième images : propagation de la segmentation aux pixels n'appartenant pas au squelette.

horizontaux, celle des traits verticaux s'en déduit aisément.

Algorithme N.4.3 : détection des traits horizontaux
Cette détection part non pas de l'image mais du gradient de cette image obtenu grâce à des filtres comme ceux du paragraphe 3.4.1, page 49. Soient $P_1 = (x_1, y)$ et $P_2 = (x_2, y)$ deux pixels de la même ligne et tels que $x_1 < x_2$. En chaque point (x, y), $g(x, y) \in \mathbb{R}^2$ désigne le gradient de l'image en ce point et $\theta(x, y)$ sa direction. On construit la suite $B = (b_1, \ldots, b_n)$ avec $n = x_2 - x_1 + 1$ définie par :

$$\forall i,\ b_i = \begin{cases} 1 & \text{si } \|g(x,y)\| \geqslant d \text{ et } \left|\theta(x,y) - \frac{\pi}{2}\right| \leqslant \theta_0 \\ 0 & \text{sinon} \end{cases} \tag{N.26}$$

d et θ_0 sont des seuils arbitraires, on pourra prendre θ_0 égal à $\frac{\pi}{16}$. L'existence permet de ne pas tenir compte des gradients trop petits dont la direction est plus le résultat d'un bruit que de l'image elle-même. On définit ensuite :

$$n(x_1, x_2, y) = \sum_{i=1}^{x_2 - x_1 + 1} \mathbb{1}_{\{b_i = 1\}} \tag{N.27}$$

$$p(x_1, x_2, y, p_0) = C_{x_2 - x_1 + 1}^{n(x_1,x_2,y)}\, p_0^{n(x_1,x_2,y)}\, (1 - p_0)^{n(x_1,x_2,y) - (x_2 - x_1 + 1)} \tag{N.28}$$

Si $p(x_1, x_2, y, p_0) < P_0$ alors le segment P_1P_2 peut être considéré comme un trait de l'image. P_0 est une probabilité qui dépend de la densité de l'image, elle sera d'autant plus faible que l'image est bruitée et que ce bruit est susceptible de créer aléatoirement des traits. Cette probabilité correspond à la probabilité qu'un bruit aléatoirement et indépendant, compte tenu de la densité de l'image, puisse être considéré comme un trait. Il va sans dire qu'une fois un segment détecté, il n'est plus la peine de considérer les sous-segments qu'il inclut.

N.4.4 Appariement

On dispose maintenant de deux ensembles de vecteurs, ceux du squelette et ceux détectés par l'algorithme N.4.3. Les vecteurs communs forment les traits des tableaux ou les soulignements. Deux segments sont considérés comme communs si : (voir figure N.18)

1. ils sont parallèles

2. la distance qui les séparent (perpendiculaires) n'est pas plus grande que $k = 3$ fois l'épaisseur du trait
3. leur longueur commune est supérieure à n fois l'épaisseur de trait ($n = 15$)

Cette étape permet de classer les vecteurs du squelette en deux ensembles, les traits T et les caractères C. Il s'agit maintenant d'extraire les caractères de la composante connexe.

Algorithme N.4.4 : extraction caractère connexe
Les ensembles T et C correspondent à l'ensemble des traits d'un tableau et l'ensemble des caractères. Ce sont deux ensembles de vecteurs de squelette mais qui peuvent être convertis en ensemble de pixels. $T = (t_1, \ldots, t_m)$ et $C = (c_1, \ldots, c_n)$. On définit la distance d'un pixel p à ces deux ensembles par :

$$d(p, T) = \min \|p - t\| \mid t \in T \tag{N.29}$$
$$d(p, C) = \min \|p - c\| \mid c \in C \tag{N.30}$$

On considère alors qu'un pixel appartient à l'ensemble des caractères ou l'ensemble des traits si :

$$p \in C \longleftrightarrow d(p, C) < d(p, T) + e_t \tag{N.31}$$
$$p \in T \longleftrightarrow d(p, T) < d(p, T) \tag{N.32}$$

Où e_t est l'épaisseur moyenne des traits du tableau. Son utilisation permet de récupérer les pixels des caractères qui se superposent à ceux du trait, ce n'est qu'une approximation de l'ensemble des pixels communs. On remarque que ces pixels appartiennent aux deux ensembles.

N.4.5 Gain de temps

Le plus long des algorithmes est la détection des segments, pour une image carrée de dimension n, son coût est $O(n^3)$. Il est possible de réduire la taille de l'image afin d'accélérer, le critère choisi est l'obtention d'un périmètre de 600 pixels. Mais pour un résultat satisfaisant, il faut un bon algorithme de réduction (conservation du gradient) ou lisser l'image avant de la réduire. Dans ce dernier cas, un lissage avec des poids gaussiens est meilleur que des poids uniformes. Cette réduction peut entraîner quelques pertes au niveau des segments détectés (voir figure N.19).

On peut également renforcer l'image réduite avec le squelette préalablement extrait et vectorisé : le squelette n'est pas ajouté à l'image mais directement sur la carte du gradient. La vectorisation permet d'éviter la déperdition de la qualité du gradient après réduction. Cette technique permet de récupérer certains segments trop fins (épaisseur = 1 pixel) à condition qu'ils soient strictement verticaux ou horizontaux sans quoi il faudrait également lisser la cadre du gradient obtenu avec le squelette.

On pourrait également n'utiliser que le squelette et réduire l'image au maximum mais cette méthode devient peu fiable. Il faudrait carrément appliquer cette méthode aux segments détectés par le squelette sans réduction de dimension.

Fig. N.19: La détection des segments n'est pas toujours parfaite, ici la ligne supérieure n'est pas toujours détectée, la ligne verticale non plus, trop fine. La détection n'est pas stable et dépend des coefficients de réduction, de ceux du lissage.

N.5 Introduction d'une forme de reconnaissance

Le principe de cette méthode de segmentation en lignes repose sur le fait que la majorité des composantes connexes sont des caractères imprimés. A partir de cet a priori, une première segmentation est faite puis affiné à partir des statistisques calculées localement sur chacune des lignes. Ceci fait apparaître un processus itératif :

1. première segmentation en lignes
2. statistiques locales
3. nouvelle segmentation en lignes : modifications locales

On peut répéter les étapes 2 et 3 jusqu'à ce que le procédé converge. Les statistiques locales proposées concernent simplement les dimensions des composantes connexes mais pourquoi ne pas faire intervenir le résultat de reconnaissance. Lorsque l'œil humain ne parvient pas à identifier une forme, il se raccroche à ce qui entoure la zone ambiguë en espérant y trouver des éléments communs. C'est ce procédé qu'il faudrait pouvoir reproduire.

N.5.1 Segmentation caractères imprimés / autres

Qu'est-ce qui distingue un caractère imprimé d'un caractère manuscrit ? Les caractères manuscrits sont souvent attachés, les composantes connexes manuscrites sont souvent plus grandes. Un caractère imprimé est plus régulier mais comment traduire sa régularité. On pourrait mesurer l'épaisseur du trait, la variance de l'estimateur de cette épaisseur mais souvent la frontière entre un caractère imprimé et un caractère manuscrit est assez floue. On pourrait étudier la distribution du rayon de courbure.

N.5.2 Calcul du rayon de courbure

Une suite discrète de pixels $L = (P_1 = (x_1, y_1), \ldots, P_n = (x_n, y_n))$ définit le contour ou le squelette d'une forme ou d'une composante connexe. On définit un nombre entier d, le rayon de courbure au pixel P_i est estimé comme la distance entre P_i et l'intersection des médiatrices des segments $[P_{i-d}, P_i]$ et $[P_i, P_{i+d}]$. On note A le milieu du segment P_1P_2, un point M appartient à la médiatrice si $\overrightarrow{AM}.\overrightarrow{P_1P_2} = 0$. On en déduit que :

$$\begin{aligned}\overrightarrow{AM}.\overrightarrow{P_1P_2} = 0 &\iff \begin{pmatrix} x - \frac{x_1+x_2}{2} \\ y - \frac{y_1+y_2}{2} \end{pmatrix} . \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} \\ &\iff 2(x_2 - x_1)x + 2(y_2 - y_1)y + y_1^2 - y_2^2 + x_1^2 - x_2^2 = 0 \end{aligned} \tag{N.33}$$

On en déduit les équations des deux médiatrices ainsi que leur intersection commune :

$$\begin{cases} 2(x_{i+d} - x_i)x + 2(y_{i+d} - y_i)y + y_i^2 - y_{i+d}^2 + x_i^2 - x_{i+d}^2 = 0 \\ 2(x_i - x_{i-d})x + 2(y_i - y_{i-d})y + y_{i-d}^2 - y_i^2 + x_{i-d}^2 - x_i^2 = 0 \end{cases} \tag{N.34}$$

$$\iff \begin{cases} 2x &= \dfrac{\left(x_i^2 - x_{i+d}^2 + y_i^2 - y_{i+d}^2\right)(y_i - y_{i-d}) - \left(x_{i-d}^2 - x_i^2 + y_{i-d}^2 - y_i^2\right)(y_{i+d} - y_i)}{(x_{i+d} - x_i)(y_i - y_{i-d}) - (x_i - x_{i-d})(y_{i+d} - y_i)} \\ 2y &= \dfrac{-\left(y_i^2 - y_{i+d}^2 + x_i^2 - x_{i+d}^2\right)(x_i - x_{i-d}) + \left(y_{i-d}^2 - y_i^2 + x_{i-d}^2 - x_i^2\right)(x_{i+d} - x_i)}{(y_{i+d} - y_i)(x_i - x_{i-d}) - (y_i - y_{i-d})(x_{i+d} - x_i)} \end{cases} \tag{N.35}$$

N.5.3 Quelques résultats

Une première suggestion consiste à choisir d égale à l'estimation de l'épaisseur du trait des lettres comme celle utilisée au paragraphe 3.4.4, page 53. Ensuite, on calcule le rayon de courbure pour chaque pixel du contour avec la valeur d. Les résultats sont illustrés par la figure N.20.

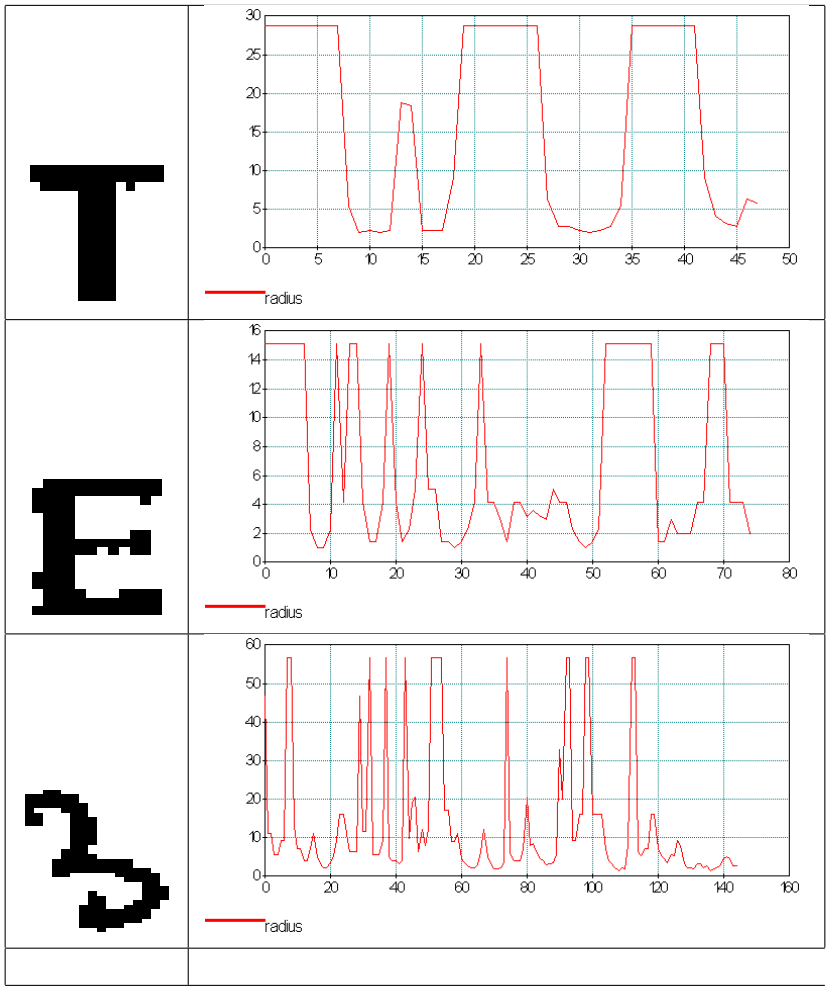


Fig. N.20: Chaque graphe représente le rayon de courbure tout au long du parcours d'une lettre. Si le rayon est infini, alors on lui attribue la valeur maximale majorée de 10 pixels. L'estimation de l'épaisseur du tracé donne 3 pixels pour les deux premières images et 4 pour la dernière.

Il semble que le graphe obtenu pour un caractère manuscrit soit plus "agité", l'entropie pourrait être un bon moyen de mesurer cette agitation. L'essentiel n'est pas que ce critère permette de discriminer dans tous les cas le style cursif du style l'imprimé mais de le permettre dans la majeure partie des cas. Ces informations, une fois agrégée au niveau d'une ligne, permettront de mieux distinguer l'imprimé du cursif au niveau de cette ligne.

N.5.4 Expériences

Cette fois-ci, la séquence des rayon de courbures est construite tout au long du contour externe d'un caractère dont la boîte englobante est de dimension $l \times h$. Lorsque le rayon est infini ou supérieur à $\max l, h$, on lui attribue la valeur $\max l, h$. On note $C = (c_1, \ldots, c_n)$, la séquence des rayons de courbure sur les pixels formés du contour et chaque rayon est calculé pour une valeur de d égale à l'épaisseur du trait. On détermine trois critères évaluer l'"agitation" de cette séquence. On note tous d'abord deux séquences :

$$C^* = \left(\frac{c_1}{\sum_{i=1}^{n} c_i}, \ldots, \frac{c_n}{\sum_{i=1}^{n} c_i} \right) \tag{N.36}$$

$$C' = \left(\frac{c_1}{\max\{c_i\}}, \ldots, \frac{c_n}{\max\{c_i\}} \right) \tag{N.37}$$

Puis les trois critères suivant :

$$s_1(C^*) = \sum_{i=1}^{n} c_i^* \ln c_i^* \tag{N.38}$$

$$s_2(C^*) = \sum_{i=2}^{n} \left| c_i^* - c_{i-1}^* \right| \tag{N.39}$$

$$s_3(C') = \sum_{i=2}^{n} \left| c_i' - c_{i-1}' \right| \tag{N.40}$$

La figure N.21 montre la distribution de ces trois critères selon le style imprimé ou cursif des caractères. L'objectif est d'essayer de construire un classifieur permettant de distinguer l'écriture cursive de l'écriture imprimée. La figure N.22 montre que les scores $s_1(C^*)$ et $s_3(C')$ sont corrélés. L'ACP ne permet non plus de conclure quant à la pertinence de ces critères.

Une faible variation ($s_3(C')$) ou une forte entropie ($s_1(C^*)$) ne signifie pas la présence de caractères imprimés. Toutefois, quelques expérience ont montré qu'une forte variation ou une faible entropie suggère fortement une composante connexe incluant un ou plusieurs caractères imprimés ou un "m" imprimé ou un style cursif.

[1]

N.6 Composantes connexes et polynômes $P(x, y) = 0$

N.6.1 Estimation

Soit $P = ((x_1, y_1), \ldots, (x_n, y_n))$ un ensemble de points pour lequel le polynôme de degré d, $P(x, y) = \sum_{0 \leqslant i+j \leqslant d} a_{ij} x_k^i y_k^j$ est nul. On cherche à estimer les coefficients de ce polynôme en minimisant l'expression suivante :

$$E = \sum_{k=1}^{n} \left(\left[\sum_{0 \leqslant i+j \leqslant d} a_{ij} x_k^i y_k^j \right] - v_k \right)^2 = 0 \tag{N.41}$$

Les variables $(v_k)_k$ sont a priori toutes nulles mais la solution la plus évidente est alors d'avoir que des coefficients nuls. On pourrait résoudre ce problème en ajoutant la contrainte $\sum_{ij} a_{ij}^2 = 1$ où on peut ajouter un point pour lequel le polynôme $p(x, y) = p_0 > 0$.

1. Annexes : voir paragraphe M, page 422

$$\forall (i,j) \neq (0,0),\ \frac{\partial E}{\partial a_{ij}} = 2\sum_{k=1}^{n}\left(x_k^i y_k^j\right)\left(\left[\sum_{0\leqslant w+z\leqslant d} a_{wz}\left(x_k^w y_k^z\right)\right] - v_k\right) = 0$$

$$\iff \forall (i,j) \neq (0,0),\ \sum_{k=1}^{n}\sum_{0\leqslant w+z\leqslant d} a_{wz}\left(x_k^w y_k^z\right)\left(x_k^i y_k^j\right) = \sum_{k=1}^{n}\left(x_k^i y_k^j\right) v_k$$

$$\iff \forall (i,j) \neq (0,0),\ \sum_{0\leqslant w+z\leqslant d} a_{wz}\sum_{k=1}^{n}\left(x_k^w y_k^z\right)\left(x_k^i y_k^j\right) = \sum_{k=1}^{n}\left(x_k^i y_k^j\right) v_k \tag{N.42}$$

On définit une fonction σ qui associe à l'ensemble à chaque entier de l'ensemble $\left\{1,\ldots,d' = \frac{(d+1)(d+2)}{2}\right\}$ un couple $(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+$ tel que $d \geqslant i+j \geqslant 0$. Cette fonction est une bijection. On définit les matrices et vecteurs suivants :

$$X = \begin{pmatrix} a_{\sigma(1)_i,\sigma(1)_j} \\ ... \\ a_{\sigma(d')_i,\sigma(d')_j} \end{pmatrix} \tag{N.43}$$

$$Z_k = \begin{pmatrix} x_k^{\sigma(1)_i} y_k^{\sigma(1)_j} \\ ... \\ x_k^{\sigma(d')_i} y_k^{\sigma(d')_j} \end{pmatrix} \tag{N.44}$$

$$A_k = V_k V_k' \tag{N.45}$$

Le système d'équations devient :

$$(N.42) \iff \left[\sum_{k=1}^{n} A_k\right] X = -\sum_{k=1}^{n} - Z_k v_k \tag{N.46}$$

Les calculs numériques posent toutefois quelques problèmes puisque les différents termes $x^i y^j$ sont d'ordre de grandeur très différents. L'ordre d'un polynôme correspond au nombre maximum de points d'intersection entre une ligne droite et la forme elle-même. Lorsque le polynôme est d'ordre supérieur à ce nombre maximum de points d'intersection, l'estimation n'est plus assez contrainte et retourne des valeurs instables pour les coefficients associés aux grandes ordres.

N.6.2 Autre approche

Cette fois-ci, on ne considère plus seulement la frontière $p(x,y) = 0$ mais l'ensemble du graphe $z = p(x,y)$. Le point (x,y) appartient à la forme si $p(x,y) > 0$. On défnit la fonction $g(x,y)$ égale à la distance du pixel (x,y) au premier pixel blanc. Cette fonction est obtenu grâce à une carte des distances. Il s'agit ici de minimiser la fonction :

$$\sum_{k=1}^{n}\left(\left[\sum_{0\leqslant i+j\leqslant d} a_{ij} x_k^i y_k^j\right] - \sqrt{g(x,y)+1}\right)^2 = 0 \tag{N.47}$$

Les problèmes rencontrés sont les mêmes mais ça permet de construire quelques images sympathiques (voir figure N.23).

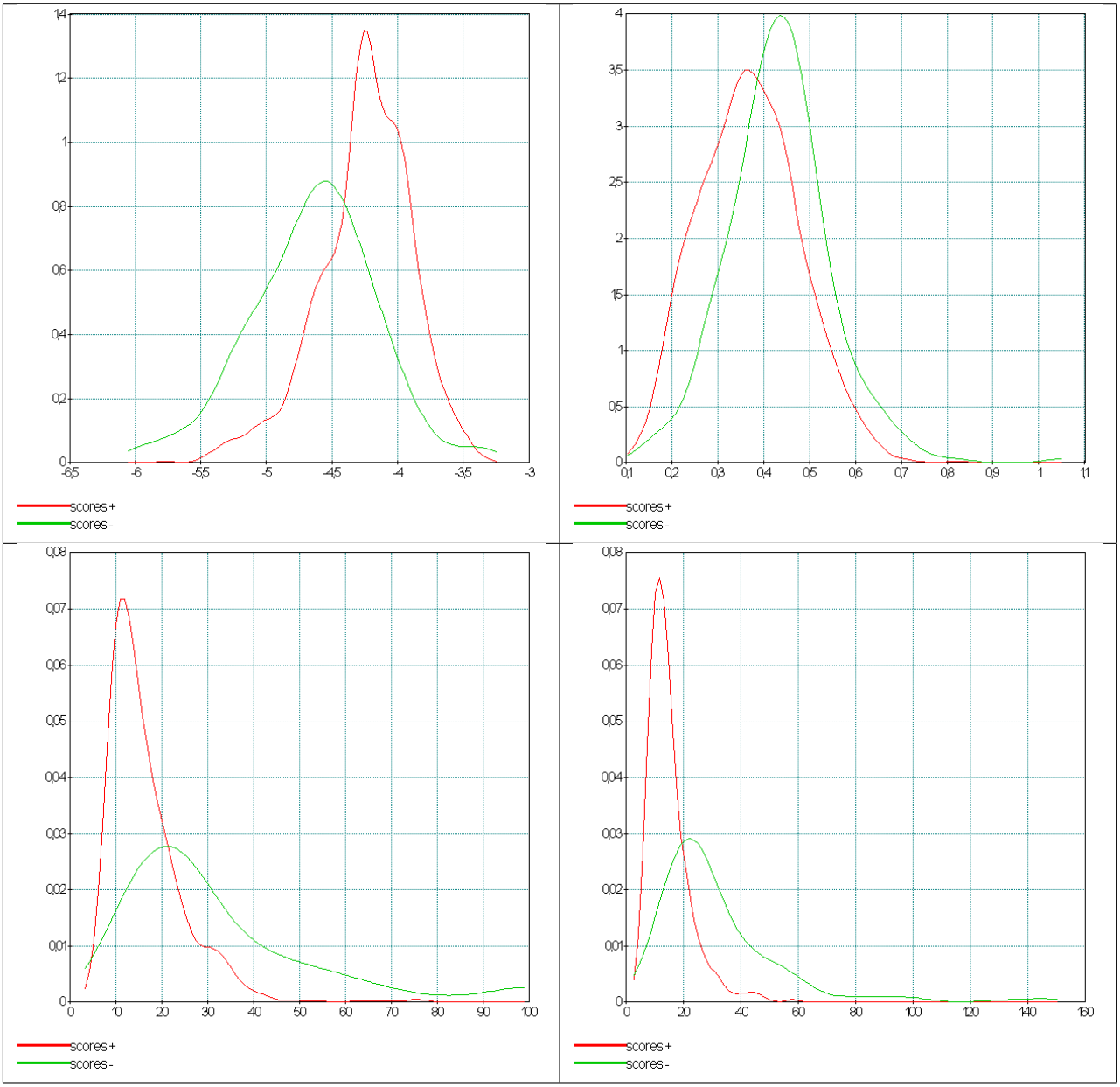


Fig. N.21: *score +* désigne la distribution des scores pour des caractères imprimés, *score* - celle des scores pour des caractères imprimés. En abscisse, on retrouve le critère $s_k(.)$, en ordonnée, la valeur de la densité en ce point. La première image désigne la distribution du score $s_1(C^*)$ défini par (N.38). La seconde image désigne la distribution du score $s_2(C^*)$ défini par (N.39). La troisième image désigne la distribution du score $s_3(C')$ défini par (N.40). Ces résultats ont été obtenus avec une base de 1698 caractères dont 332 manuscrits. La dernière image le critère $s_3(C')$ pour une valeur de d constante et égale à 1. Elle montre que ce critère, comme les autres d'ailleurs, sont peu sensibles au paramètre d.

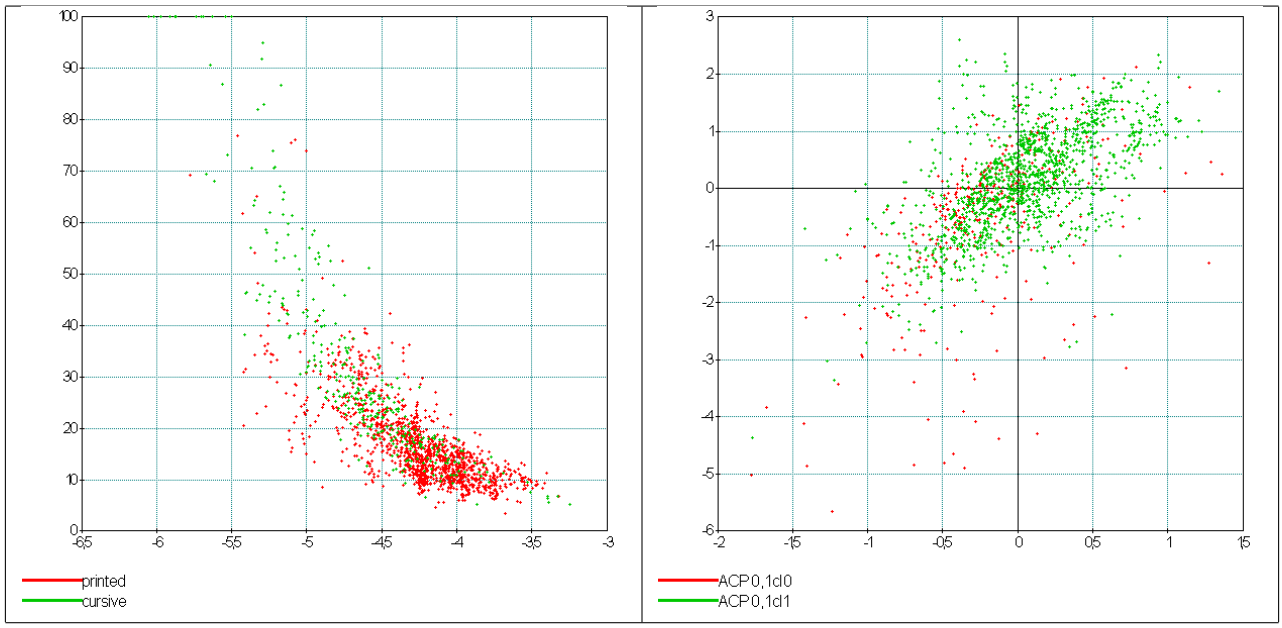


Fig. N.22: La première image montre la corrélation entre les critères $s_1(C^*)$ et $s_1(C^*)$. L'ACP ne permet pas non de conclure quant à la pertinence de ces trois critères.

Fig. N.23: $-2,71415271567460x - 0,17513551554947y - 0,45764014382701x^2 + 0,59165264727629xy - 0,35689679679945y^2 + 6,3675891583915x^3 - 0,27804657119979x^2y + 1,7191284357251xy^2 + 0,34717141073779y^3 + 2,2097205099161x^4 - 0,92011576789963x^3y - 0,53771900512184x^2y^2 - 0,44132587775827xy^3 + 0,41155517559946y^4 - 3,8251599582292x^5 + 0,33769625788991x^4y - 2,5627669760007x^3y^2 + 0,14610259793954x^2y^3 - 0,75737548063447xy^4 - 0,29843149760046y^5 - 1,5823152685729x^6 + 0,49203716664659x^5y + 0,068201171622675x^4y^2 + 0,30377704699013x^3y^3 - 0,10672683086398x^2y^4 + 0,24231302290055xy^5 - 0,22016041531788y^6 + 1,0032037406334x^7 - 0,12540223425241x^6y + 1,0137511140285x^5y^2 - 0,083905588250162x^4y^3 + 0,46096219376055x^3y^4 - 0,046331597291701x^2y^5 + 0,18750499181024xy^6 + 0,116441898178197y^7 + 0,43072233684903x^8 - 0,11500481291936x^7y + 0,091401140595046x^6y^2 - 0,13591801843544x^5y^3 + 0,055346171982881x^4y^4 - 0,026800706915417x^3y^5 + 0,063910166291767x^2y^6 - 0,084723759839934xy^7 + 0,052726689358906y^8 - 0,12749361391745x^9 + 0,016003250119303x^8y - 0,13994797779189x^7y^2 + 0,025699808739355x^6y^3 - 0,11371439591465x^5y^4 + 0,0039637614703148x^4y^5 - 0,044709857700699x^3y^6 + 0,0092461484289719x^2y^7 - 0,020866891196708xy^8 - 0,020397084793952y^9 - 0,049668949491666x^{10} + 0,013606483334884x^9y - 0,032077882808888x^8y^2 + 0,019740770244965x^7y^3 - 0,0015884085631117x^6y^4 + 0,012894943312007x^5y^5 - 0,0250273235699x^4y^6 - 0,0011026418339998x^3y^7 - 0,0050096439432752x^2y^8 + 0,01447844493903xy^9 - 0,0055661075545004y^{10} + 0,0065648114799028x^{11} - 0,00049572580859945x^{10}y + 0,0055139726435325x^9y^2 - 0,0025515252117933x^8y^3 + 0,0098785305378567x^7y^4 - 0,00015443463752532x^6y^5 + 0,0025942501837917x^5y^6 - 0,0003530735867445x^4y^7 + 0,0025919898482227x^3y^8 - 0,00063496798195889x^2y^9 + 0,00074416655667094xy^{10} + 0,0013041723471905y^{11} + 0,0019706372604758x^{12} - 0,00072026319665604x^{11}y + 0,0029775119756249x^{10}y^2 - 0,0007604634861955 8x^9y^3 - 0,00060076338485865x^8y^4 - 0,0013324531333535x^7y^5 + 0,0016888185054 79x^6y^6 - 0,00016733728791066x^5y^7 + 0,0014925727209504x^4y^8 + 0,00014742703215478x^3y^9 - 6,0901214722686e - 005x^2y^{10} - 0,00091525125573819xy^{11} + 0,00021417042222183y^{12}$

Bibliographie

---

[1] Reconnaissance et modélisation

[Amit1997] Y. Amit, D. Geman, *Shape Quantization and Recognition with Randomized Trees*, Neural Computation 9 (1997), pp 1545-1588

[Augustin2001] E. Augustin, *Reconnaissance de mots manuscrits par systèmes hybrides réseaux de neurones et modèles de Markov cachés*, Thèse de l'Université Paris V (2001)

[Balasubramanian1993] V. Balasubramanian, *Equivalence and Reduction of Hidden Markov Models*, Massachussetts Institute of Technology Artificial Intelligence Laboratory A.I. Technical Report No. 1370 (1993)

[Baum1968] L. E. Baum L.E., G. R. Sell, *Growth transformation for functions on manifolds*, Pac. J. Math. 27 (1968), pp 211-227

[Baum1972] L. E. Baum, *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process*, Inequalities 3 (1972), pp 1-8

[Bengio1992] Y. Bengio, R. De Mori, G. Flammia, R. Kompe, *Global Optimization of a Neural Network-Hidden Markov Model Hybrid*, IEEE Transactions on Neural Networks 3(2) (1992), pp 252-259

[Bengio1996] Y. Bengio, P. Frasconi, *Input/Output HMMs for sequence processing*, IEEE Transactions on Neural Network, 7(5) 1231 (1996), pp 1249-

[Bicego2003] Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *A sequential pruning strategy for the selection of the number of states in hidden Markov Models*, Pattern Recognition Letters 24 (2003), pp 1395-1407

[Bishop1995] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press (1995)

[Bottou1991] L. Bottou, *Une approche théorique de l'apprentissage connexionniste, Application à la reconnaissance de la parole*, Thèse de l'Université dfe Paris Sud, Centre d'Orsay (1991)

[Bunke1995] H. Bunke, M. Roth, E. G. Schukat-Talamazzini, *Off-line cursive handwriting recognition using hidden Markov models*, Pattern Recognition 28 (1995), pp 1399-1413

[Burges1998] C. J. C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery 2(2) (1998), pp 955-974

[Cai2002] Jinhai Cai, Zhi-Qiang Liu, *Pattern recognition using Markov random field models*, Pattern Recognition 35 (2002), pp 725-733

[Cottrel1995] M. Cottrel, B. Girard, M. Mangeas, C. Muller, *Neural modeling for time series : a statistical stepwise methode for weight elimination*, IEEE Transaction On Neural Networks 6(6) (1995)

[Dempster1977] A. P. Dempster, N. M. Laird, D. B. Rubin, *Maximum-Likelihood from incomplete data via the EM algorithm*, Journal of Royal Statistical Society B 39 (1977), pp 1-38

[Eickeler1998] S. Eickeler, S. Müller, G. Rigoll, *Improved Face Recognition Using Pseudo-2D Hidden Markov Models*, Workshop on Advances in Facial Image Analysis and Recognition Technology in conjunction with 5th European Conference on Computer Vision (ECCV'98), Freiburg, Germany (1998)

[Kamp1985] Y. Kamp, *State Reduction in Hidden Markov Chains Used for Speech Recognition*, IEEE Transactions of Acoustics, Speech and Signal Processing 33 (5) (1985), pp 1138-1145

[Knerr2000] S. Knerr, P. Lallican, C. Viard-Gaudin, *From Off-Line to On-Line handwriting Recognition*, International Workshop on Frontiers in Handwriting Recognition, IWFHR'2000, Amsterdam, The Netherlands (2000), pp 303-312

[Knerr2001] S. Knerr, Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, *An offline cursive handwritten word recognition system*, In Proceedings of IEEE Region 10 Conference (2001)

[Koerich2002b] A. L. Koerich, R. Sabourin, Y. Leydier, C. Y. Suen, *A Hybrid Large Vocabulary Handwritten Word Recognition System using Neural Networks with Hidden Markov Models*, 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR'8) (2002), pp 99-104

[Khorsheed2003] M. S. Khorsheed, *Recognising handwritten Arabic manuscripts unsing a single hidden Markov model*, Pattern Recognition Letters 24 (2003), pp 2235-2242

[Krogh1994] A. Krogh, *Hidden Markov Models for Labeled Sequency*, Proceeding of the 12th IAPR ICPR'94 (1994), pp 140-144

[Kuo1994] S. Kuo, E. Agazzi, *Keyword Spotting in Poorly Printed Documents Using Pseudo 2-D Hidden Markov Models*, IEEE Transactions on Pami 16(8) (1994), pp 842-848

[Lallican1999] P. M. Lallican, *Reconnaissance de l'écriture manuscrite hors-ligne : utilisation de la chronologie restaurée du tracé*, Thèse de l'Université de Nantes IRESTE (1999)

[Lecolinet1990] E. Lecolinet, *Segmentation d'images de mots manuscrits*, Thèse de l'Université Paris VI (1990)

[Lecolinet1996] E. Lecolinet, *A Survey of Methods and Strategies in Character Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 18(7) (1996), pp 690-706

[LeCun1998] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE 86 (11) (1998), pp 2278-2324

[Lempel1978] A. Lempel, J. Ziv, *Compression of individual sequences via variable rate coding*, IEEE, Transactions on Information Theory IT-24 (1978), pp 530-536

[Levinson1983] S. E. Levinson, L. R. Rabiner, N. M. Sondhi, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, Bell System Technical Journal 52 (4) (1983), pp 257-285

[Mohri1996] M. Mohri, *Finite-state transducers in language and speech processing*, Computational Linguistics 20(1) (1996), pp 1-33

[Niles1990] L. T. Niles, *Modelling and Learning in Speech Recognition : the relationship between stochastic pattern classifiers and neural networks*, Technical Report LEMS-79, Brown University (1990)

[Rabiner1986] L. R. Rabiner, B. H. Juang, *An introduction to hidden Markov models*, IEEE ASSP Magazine January (1986), pp 4-15

[Riis1998] S. K. Riis, *Hidden Markov models and neural networks for speech recognition*, Thèse de l'Université Technique du Danemark (1998)

[Saon1997] G. Saon, A. Belaïd, *High Performance Unconstrained Word Recognition System Combining HMMs and Markov Random Field*, International Journal on Pattern Recognition and Artificial Intelligence (IJPRAI) Special Issue on Automatic Bankcheck Processing, S. Impedovo Ed. vol. 11, n. 5 (1997), pp 771-788

[Saporta1990] Gilbert Saporta, *Probabilités, analyse des données et statistique*, Editions Technip (1990)

[Sayre1973] K. Sayre, *Machine recognition of handwritten words : a project report*, Pattern Recognition 5(3) (1973), pp 213-228

[Schenkel1995] M. Schenkel, I. Guyon, D. Henderson, *On-line cursive script recognition using time delay neural networks and hidden Markov models*, Machine Vision and Applications (1995), pp 215-223

[Senior1994] A. Senior, *Off-line Cursive Handwriting Recognition using Recurrent Neural Network*, PHD Thesis in Trinity Hall, Canmbridge, England (1994)

[Senior1998] A. Senior, *An off-line cursive handwriting recognition system*, IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3) (1998), pp 309-321

[Steinherz1999] T. Steinherz, E. Rivlin, N. Intrator, *Offline cursive script word recognition - a survey*, Internation Jounral on Document Analysis and Recognition (IJDAR) Abstract Volume 2 Issue 2/3 (1999), pp 90-110

[Simon1992] J. C. Simon, O. Baret, *Cursive words recognition*, From Pixel to Features III : Frontiers in Handwriting Recognition, S. Impedovo and J.C. Simon (eds.) Elsevier Science Publishers B.V. (1992), pp 241-260

[Vapnik1979] V. N. Vapnik, *Estimation of dependances based on empirical data*, (en russe), Nauka, Moscow, 1979, traduction anglaise : Springer-Verlag, New-York (1982) (1979)

[Verma2004] Brijest Verma, Michael Blumenstein, Moumita Glosh, *A novel approach for structural feature extraction : Contour vs direction*, Pattern Recognition Letters 25 (2004), pp 975-988

[Vinciarelli2002] A. Vinciarelli, *A survey on off-line Cursive Word Recognition*, Pattern Recognition 35 (2002), pp 1433-1446

[Ziv1992] J. Ziv, N. Merhav, *Estimating the number of states of a finite state source*, IEEE Transactions on Information Theory 38(1) (1992), pp 61-65

[2] Traitement de l'image

[Abuhaiba1996] I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, *Processing of binary images of handwritten text documents*, Pattern Recognition 29 (1996), pp 1161-1177

[Augustin2001] - cité aussi page 453

E. Augustin, *Reconnaissance de mots manuscrits par systèmes hybrides réseaux de neurones et modèles de Markov cachés*, Thèse de l'Université Paris V (2001), pp 0-

[Baret1991] O. Baret, *Régularités, singularités de représentations et leur complémentarité : application à la reconnaissance de l'écriture manuscrite non contrainte*, Thèse de l'Université Paris VI (1991)

[Bloomberg1995] D. S. Blommberg, G. E. Kopec, L. Dasari, *Measuring document image skew and orientation*, Vincent, L.M., Baird, H.S. (Eds), Proc. SPIE :Document Recogntiion II, San Jose, California 2422 (1995), pp 302-316

[Bozinovic1989] R. M. Bozinovic, S. N. Srihari, *Off-line cursive script word recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence 11(1) (1989), pp 68-83

[Bresenham1965] J. E. Bresenham, *Algorithm for computer control of a digital plotter*, IBM System Journal 4(1) (1965), pp 25-30

[Bresenham1977] J. E. Bresenham, *A linear algorithm for incremental digital display of circuler arcs*, CGIP 20(2) (1977)

[Cao2003] Yang Cao, Shuhua Wang, Heng Li, *Skew detection and correction in document images based on straiht-line detection*, Pattern Recognition Letters 24 (2003), pp 1871-1879

[Cheng2004] Zhiguo Cheng, Mang Chen, Yuncai Liu, *A robust algorithm for image principal curve detection*, Pattern Recognition Letters 25 (2004), pp 1303-1313

[Côté1997] M. Côté, M. Cheriet, E. Lecolinet, C. Y. Suen, *Détection des Lignes de Référence de Mots Cursifs à l'aide de l'entropie*, Les techniques de l'I.A. appliquées aux technologies de l'information, Richard Lepage and Rita Noumeir (Eds.), Cahiers Scientifiques de l'ACFAS 90 (1997), pp 184-193

[Desolneux2000] Agnès Desolneux, Lionel Moisan, Jean-Michel Morel, *Meaningful Alignments*, International Journal of Computer Vision 40(1) (2000), pp 7-23

[Desolneux2002] Agnès Desolneux, Lionel Moisan, Jean-Michel Morel, *Gestalt theory and Computer Vision*, Publication du Centre de Mathématiques et de Leurs Applications, disponible à l'adresse http ://www.cmla.ens-cachan.fr/Cmla/Publications/ Nž 2002-06 (2002)

[Desolneux2003] Agnès Desolneux, Lionel Moisan, Jean-Michel Morel, *A Grouping Principle and Four Applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence 25(4) (2003), pp 508-513

[Dijkstra1971] E. W. Dijkstra, *Hierarchical ordering of sequential processes*, Acta Informatica 1,2 (1971), pp 115-138

[Dupré2000] X. Dupré, *Reconnaissance de mots-clé dans un document manuscrit*, Mémoire de DEA de l'Université Paris VI (2000)

[Elnagar2003] Ashraf Elnagar, Reda Alhajj, *Segmentation of connected handwritten numeral strings*, Pattern Recognition 36 (2003), pp 625-634

[Gatos1997] B. Gatos, N. Papamarkos, C. Chamzas, *Skew detection and text line position, determination in digitized documents*, Pattern Recognition 30 (1997), pp 1505-1519

[Hennig2002] A. Hennig, N. Sherkat, *Exploiting zoning based on approximation splines in cursive script recognition*, Pattern Recognition 35 (2002), pp 445-454

[Hwang1998] Wen L. Wang, Fu Chang, *Character extraction from documents using wavelets maxima*, Image and Vision Computing 16 (1998), pp 307-315

[Lecolinet1991] E. Lecolinet, J. P. Crettez, *A grapheme-bases segmentation technique for cursive script recognition*, Actes de ICDAR'91 (International Conference on Document Analysis and Recognition), Saint-Malo 740 (1991), pp 748-

[Kapoor2004] Rajiv Kapoor, Deepak Bagai, T.S. Kamal, *A new algorithm for skew detection and correction*, Pattern Recognition Letters 25 (2004), pp 1215-1229

[Kim1997] G. Kim, V. Govindaraju, *A lexicon driven approach tohandwritten word recognition for real-time applications*, Transactions on Pattern Analysis and Machine Intelligence 19(4) (1997), pp 366-379

[Knerr1997] S. Knerr, A. Anisimov, O. Baret, N. Gorski, D. Price, J. C. Simon, *The A2iA intercheque system : Courtesy amount and legal amount recognition for french checks*, S. Impedovo, P.S.P. Wang and H. Bunke, editors, Automatics Bankcheck Processing, vol 28 of Series in Machine Perception and Artificial Intelligence, World Scientific (1997), pp 43-86

[Knerr2001] - cité aussi page 454

S. Knerr, Y. H. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, *An offline cursive handwritten word recognition system*, Proceedings of IEEE Region 10 Conference (2001), pp 0-

[Kwon2004] Soon H. Kwon, *Threshold selection based on cluster analysis*, Pattern Recognition Letters 25 (2004), pp 1045-1050

[Lu1996] Yi Lu, M. Shridhar, *Character segmentation in handwritten words - an overview*, Pattern Recognition 29 (2003), pp 77-96

[Lu2003] Yue Lu, Chew Lim Tan, *A nearest-neighbor chain based approach to skew estimation in document images*, Pattern Recognition Letters 24 (2003), pp 2315-2323

[Madhvanath1999] S. Madhvanath, V. Govindaraju, *Local reference lines for handwritten phrase recognition*, Pattern Recognition 32 (1999), pp 2021-2028

[Madhvanath2001] S. Madhvanath, V. Krpasundar, V. Govindaraju, *Syntactic methodology of pruning large lexicons in cursive script recognition*, Pattern Recognition 34 (2001), pp 37-46

[Pal1996] U. Pal, B. B. Chaudhuri, *An improved document skew angle estimation technique*, Pattern Recognition Letters 17 (1996), pp 899-904

[Pal2001] U. Pal, B. B. Chaudhuri, *Machine-printed and hand-written text lines identification*, Pattern Recognition Letters 22 (2001), pp 431-441

[Pal2003] U. Pal, A. Belaïd, Ch. Choisy, *Touching numeral segmentation using water reservoir concept*, Pattern Recognition Letters 24 (2003), pp 261-272

[Prewitt1970] J. M. S. Prewitt, *Object enhancement and extraction*, Picture Processing and Psychopictorics, , B.S. Likin and A. Rosenfeld Academic Press (1970), pp 75-149

[Slavik2000] P. Slavik, V. Govindaraju, *An Overview of Run-Length Encoding of Handwritten Word Images*, Technical Report at Department of Computer Science and Engineering, SUNY at Buffalo (2000)

[Simon1992] - cité aussi page 455

J. C. Simon, O. Baret, *Cursive words recognition*, From Pixel to Features III : Frontiers in Handwriting Recognition, S. Impedovo and J.C. Simon (eds.) Elsevier Science Publishers B.V. (1992)

[Vinciarelli2000] A. Vinciarelli, J. Luettin, *A new normalization technique for cursive handwritten words*, IDIAP RR, 00-32 (2000)

[Wang1997] Jiren Wang, Maylor K. H. Leung, Sui Cheung Hui, *Cursive word reference line detection*, Pattern Recognition 30 (1997), pp 503-511

[Wang1999] Jiangua Wang, Hong Yan , *Mending broken handwriting with a macrostructure analysis method to improve recognition*, Pattern Recognition Letters 20 (1999), pp 855-864

[Whichello1996] Adrian P. Whichello, Hong Yan , *Linking broken character borders with variable size masks to improve recognition*, Pattern Recognition 29 (1996), pp 1429-1435

[Yanikoglu1998] B. Yanikoglu, P. A. Sandon, *Segmentation of off-line cursive handwriting using linear programming*, Pattern Recognition 31 (1998), pp 1825-1833

[You2003] Daekun You, Gyeonghwan Kim, *An efficient approach for slant correction of handwritten Korean strings based on structural properties*, Pattern Recognition Letters 24 (2003), pp 2093-2101

[Zhao2000] Zhao Xuesheng, CHEN Jun and Wang Jinzhuang, *Overlay Generalization of Polygons based on Fuzzy Voronoi*, Journal of China University of Mining and Technology 10(2) (2000), pp 120-125

[3] Reconnaissance statistique

[Abou-Mustafa2004] K. T. Abou-Mustafa, M. Cheriet, C. Y. Suen, *On the structure of hidden Markov models*, Pattern Recognition Letters 25 (2004), pp 923-931

[Akaike1974] H. Akaike, *A new look at the statistical model identification*, IEEE Transactions on Automatic Control AC-19 (1974), pp 716-723

[Amit1997] - cité aussi page 453

Y. Amit, D. Geman, *Shape Quantization and Recognition with Randomized Trees*, Neural Computation 9 (1997), pp 1545-1588

[Augustin2001] - cité aussi page 453

E. Augustin, *Reconnaissance de mots manuscrits par systèmes hybrides réseaux de neurones et modèles de Markov cachés*, Thèse de l'Université Paris V (2001), pp 0-

[Balasubramanian1993] - cité aussi page 453

V. Balasubramanian, *Equivalence and Reduction of Hidden Markov Models*, Massachussetts Institute of Technology Artificial Intelligence Laboratory, A.I. Technical Report No. 1370 (1993), pp 0-

[Barandela2003] R. Barandela, J.S. Sanchez, V. Garcia, E. Rangel], *Strategies for learning in class imbalance problems*, Pattern Recognition 36 (2003), pp 849-851

[Bengio1995] Y. Bengio, P. Frasconi, *Diffusion of credits in Markovian model.*, Neural Information Process 7 (1995), pp 1251-1254

[Bengio1996] - cité aussi page 453

Y. Bengio, P. Frasconi, *Input/Output HMMs for sequence processing*, IEEE Transactions on Neural Network 7(5) (1996), pp 1231-1249

[Bicego2003] - cité aussi page 453

Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *A sequential pruning strategy for the selection of the number of states in hidden Markov Models*, Pattern Recognition Letters 24 (2003), pp 1395-1407

[Bicego2004] Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *Similarity-based classification of sequences using hidden Markov models*, Pattern Recognition 37 (2004), pp 2281-2291

[Chen1994] Mou-Yen Chen, Amlan Kundu, Jian Zhou, *Off-line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network*, IEEE Transations on Pattern Analysis and Machine Intelligence 16(5) (1994), pp 481-496

[Chen2003] G. Y. Chen, T. D. Bui, A. Kryzak, *Contour-based handwritten recognition using multiwavelets and neural networks*, Pattern Recognition 36 (2003), pp 1597-1604

[Choi2003b] Euisun Choi, Chulhee Lee, *Feature extraction based on the Bhattacharyya distance*, Pattern Recognition 36 (2003), pp 1703-1709

[ChoiH2003] Hyunil Choi, Sung-Jung Cho, Jin H. Kim, *Generation of Handwriting Characters with Bayesian network based On-Line Handwriting Recognizers*, Internation Conference on Pattern Analysis and Recognition (2003), pp 995-999

[Connell2000] S. D. Connell, *On-Line handwriting recognition using multiple pattern class models*, Thesis, Michigan State University (2000)

[Datta1997] Amitava Datta, S.K. Parui, *Skeletons from dot patterns : A neural network approach*, Pattern Recognition Letters 18 (1997), pp 335-342

[Davies1979] D. L. Davies, D. W. Bouldin, *A cluster Separation Measure*, IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 1(2) (1979)

[Dempster1977] - cité aussi page 453

A. P. Dempster, N. M. Laird, D. B. Rubin, *Maximum-Likelihood from incomplete data via the EM algorithm*, Journal of Royal Statistical Society B 39 (1977)

[Dong2003] Ming Dong, Ravi Kothari, *Feature subset selection using a new definition of classifiability*, Pattern Recognition Letters 24 (2003), pp 1215-1225

[Dupré2002] X. Dupré, *Reconnaissance de l'écriture manuscrite*, Rapport interne du laboratoire SIP-CRIP5 de l'Université Paris V (2002)

[Dupré2004] X. Dupré, E. Augustin, *Hidden Markov Models for Couples of Letters Applied to Handwriting Recognition*, Pattern Recognition, 17th International Conference on (ICPR'04) 2 (2004), pp 618-621

[Durand2003] Jean-Baptiste Durand, *Modèles à structure cachée : inférence, sélection de modèles et applications*, Thèse de l'Université Joseph Fourier (2003)

[Fukunaga1990] K. Fukunuga, *Introduction to Statistical Pattern Recognition*, Academic Press, Boston, M.A. (1990)

[Glendinning2003] R. H. Glendinning, R. A. Herbert, *Shape classification using smooth principal components*, Pattern Recognition Letters 24 (2003), pp 2021-2030

[Günter2003] S. Günter, H. Bunke, *HMM-based handwritten word recognition : on the optimization of the number of states, training iterations and Gaussian components*, Pattern Recognition 37 (2003), pp 2069-2079

[Heikkilä2004] Janne Heikkilä, *Pattern matching with affine moment descriptors*, Pattern Recognition 37 (2004), pp 1825-1834

[Herbin2001] M. Herbin, N. Bonnet, P. Vautrot, *Estimation of the number of clusters and influence zones*, Pattern Recognition Letters 22 (2001), pp 1557-1568

[Hoti2004] Fabian Hoti, Lasse Holmström, *A semiparametric density estimation approach to pattern classification*, Pattern Recognition 77 (2004), pp 409-419

[Hu1961] M. Hu, *Pattern recognition by moments invariants*, Proc IRE 49, 1428 (1961)

[Hu1962] M. Hu, *Visual pattern recognition by moments invariants*, IRE Transactions on Informations Theory 8 (1962), pp 179-187

[ICDAR2003] ICDAR, *http ://www.essex.ac.uk/ese/icdar2003/index.htm ou http ://www.iam.unibe.ch/ zimmerma/iamdb/iamdb.html*, Seventh International Conference on Document Analysis and Recognition (2003)

[Jin2004] Liu Jin, Zhang Tianxu, *Fast algorithm for generation of moment invariants*, Pattern Recognition 37 (2004), pp 1745-1756

[Kamp1985] - cité aussi page 454

Y. Kamp, *State Reduction in Hidden Markov Chains Used for Speech Recognition*, IEEE Transactions of Acoustics, Speech and Signal Processing, Vol. ASSP-33 5 (1985), pp 1138-1145

[Kohonen1997] T. Kohonen, *Self-Organizing Map*, Springer (1997)

[Khorsheed2003] - cité aussi page 454

M. S. Khorsheed, *Recognising handwritten Arabic manuscripts unsing a single hidden Markov model*, Pattern Recognition Letters 24 (2003)

[Kuhl1982] F. P. Kuhl, C. R. Giardina, *Elliptic Fourier features of a closed contour*, Computer Vision Graphics Image Processing 18 (1982), pp 236-258

[Levinson1983] - cité aussi page 454

S. E. Levinson, L. R. Rabiner, N. M. Sondhi, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, Bell System Technical Journal 52(4) (1983), pp 257-285

[Li1992] Y. Li, *Reforming the theory of invariant moments for pattern recognition*, Pattern Recognition 25 (1992), pp 723-730

[Li2004] Jie Li, Jiaxin Wang, Yannan Zhao, Zehong Yang, *Self-adaptative of hidden Markov Models*, Pattern Recognition Letters 25 (2004), pp 197-210

[LiuCL2003] Chang-Lin Liu, Kazuki Nakashima, Hiroshi Sako, Hiromichi Fujisawa, *Handwritten digit recognition : benchmarking of state-of-the-art techniques*, Pattern Recognition 36 (2003), pp 2271-2285

[Mitchell1995] C. D. Mitchell, M. P. Harper and L. H. Jamieson, *On the Complexity of Explicit Duration HMMs*, IEEE Transactions on Speech and Audio Processing 3(3) (1995), pp 213-217

[Oliveira2002] Jr J. J. Oliveira, J. M. de Carvalho, C. O. A. Freitas, R. Sabourin, *Feature Sets Evaluation for Handwritten Word Recognition Using a Baseline System*, 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR'8), Niagara-on-the-Lake (2002), pp 446-451

[Rabiner1986] - cité aussi page 454

L. R. Rabiner, B. H. Juang, *An introduction to hidden Markov models*, IEEE ASSP Magazine January (1986), pp 4-15

[Ruberto2004] Cecilia Di Ruberto, *Recognition of shapes by attibuted skeletal graphs*, Pattern Recognition 37 (2004), pp 21-31

[Saporta1990] - cité aussi page 454

Gilbert Saporta, *Probabilités, analyse des données et statistique*, Editions Technip (1990), pp 0-

[Schwartz1978] G. Schwartz, *Estimation the dimension of a model*, Annals of Statistics 6 (2) (1978), pp 461-464

[Sederberg1992] T. W. Sederberg, E. Greenwood, *A physically based approach to 2-d shape blanding*, Computer Graphics (Proc. SIGGRAPH) 26 (1992), pp 25-34

[Senior1994] - cité aussi page 455

A. Senior, *Off-line Cursive Handwriting Recognition using Recurrent Neural Network*, PHD Thesis in Trinity Hall, Canmbridge, England (1994), pp 0-

[Senta1986] E. Senta, *Nonnegative matrices and Markov chains*, Springer, New-York (1986)

[Shen1999] Dinggang Shen, Horace H. S. Ip, *Discriminative wavelet shape descriptors for recognition of 2-D patterns*, Pattern Recognition 32 (1999), pp 151-165

[Silverman1986] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability, Chapman and Hall, London 26 (1986)

[Tao2001] Yu Tao, Ernest C.M. Lam, Yuan Y. Tang, *Feature extraction using wavelet and fractal*, Pattern Recognition Letters 22 (2001), pp 271-287

[Tappert1984] C.C. Tappert, *Adaptative On-Line Handwriting Recognition*, Proc. 7th Int. Conf. on Pattern Recognition, Montreal, Canada, 99 (1984), pp 1004-1007

[Trier1996] Øivind Due Trier, Anil K. Jain, Torfinn Taxt, *Feature extraction methods for character recognition - a survey*, Pattern Recognition 29 (1996), pp 641-662

[Uchida2003] Seiichi Uchida, Hiroaki Sakoe, *Eigen-deformation for elastic matching based handwritten character recognition*, Pattern Recognition 36 (2003), pp 2031-2040

[Verma2004] - cité aussi page 455

Brijest Verma, Michael Blumenstein, Moumita Glosh, *A novel approach for structural feature extraction : Contour vs direction*, Pattern Recognition Letters 25 (2004), pp 975-988

[Waard1995] W. P. de Waard, *An optimised distance method for character recognition*, Pattern Recognition Letters 16 (1995), pp 499-506

[Wang2000] Xian Wang, Venu Govindaraju, Sargur Srihari, *Holistic recognition of handwritten character pair*, Pattern Recognition 33 (2000), pp 1967-1973

[Wong1995] Wai-Hong Wong, Wan-Chi Siu, Kin-Man Lam, *Generation of moment invariants and their uses for character recognition*, Pattern Recognition Letters 16 (1995), pp 115-123

[Wunsch1995] Patrick Wunsch, Andrew F. Laine, *Wavelet descriptors for multiresolution recognition of handprinting characters*, Pattern Recognition 28 (1995), pp 1237-1249

[Ziv1992] - cité aussi page 455

J. Ziv, N. Merhav, *Estimating the number of states of a finite state source*, IEEE Transactions on Information Theory 38(1) (1992), pp 61-65

[ZhangD2004] Dengsheng Zhang, Guojun Lu, *Review of shape representation and description techniques*, Pattern Recognition 37 (2004), pp 1-19

[4] Décision

[Augustin2001] - cité aussi page 453

E. Augustin, *Reconnaissance de mots manuscrits par systèmes hybrides réseaux de neurones et modèles de Markov cachés*, Thèse de l'Université Paris V (2001), pp 0-

[Dupré2003] X. Dupré, *Optimization of cursive words recognition and nearest neighbors search in metric spaces*, International Conference on Image and Signal Processing (ICISP), Agadir, Morroco (2003)

[Günter2004] Simon Günter, Horst Bunke, *Feature selection algorithm for the generation of multiple classifier systems and their application to handwritten word recognition*, Pattern Recognition Letters 25 (2004), pp 1323-1336

[Koerich2002a] A. L. Koerich, R. Sabourin, C. Y. Suen, *Fast two-level viterbi search algorithm for unconstrained handwriting recognition*, Proc. 27th International Conference on Acoustics Speech and Signal Processing, Orlando, USA (2002)

[Koerich2002b] - cité aussi page 454

A. L. Koerich, R. Sabourin, Y. Leydier, C. Y. Suen , *A Hybrid Large Vocabulary Handwritten Word Recognition System using Neural Networks with Hidden Markov Models*, 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR'8), Niagara-on-the-Lake (2002)

[Lin2003] Xiafon Lin, Sherif Yacoub, John Burns, Steven Simske, *Performance analysis of pattern classifier combination by plurality voting*, Pattern Recognition Letters 24 (2003), pp 1959-1969

[Madhvanath2001] - cité aussi page 456

S. Madhvanath, V. Krpasundar, V. Govindaraju, *Syntactic methodology of pruning large lexicons in cursive script recognition*, Pattern Recognition 34 (2001), pp 37-46

[Senior1994] - cité aussi page 455

A. Senior, *Off-line Cursive Handwriting Recognition using Recurrent Neural Network*, PHD Thesis in Trinity Hall, Canmbridge, England (1994), pp 0-

[Senior1998] - cité aussi page 455

A. Senior, *An off-line cursive handwriting recognition system*, IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3) (1998), pp 309-321

[Wunsch1995] - cité aussi page 460

Patrick Wunsch, Andrew F. Laine, *Wavelet descriptors for multiresolution recognition of handprinting characters*, Pattern Recognition 28 (1995), pp 1237-1249

[5] Nouveaux enjeux

[Koch2005] G. Koch, L. Heutte, T. Paquet, *Automatic extraction of numerical sequences in handwritten mail documents*, Pattern Recognition Letters 26 (2005), pp 1118-1127

[Likforman1995] L. Likforman, C. Faure, *Une méthode de résolution des conflits d'alignements pour la segmentation des documents manuscrits*, Traitement du signal 12 (1995), pp 541-549

[6] Squelettisation

[Abuhaiba1996] - cité aussi page 455

I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, *Processing of binary images of handwritten text documents*, Pattern Recognition 29 (1996), pp 1161-1177

[Arcelli1985] C. Arcelli, B. Sanniti di Baja, *A width-independent fast thinning algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence 7(4) (1985), pp 463-474

[Attali1995] D. Attali, *Squelette et Graphes de Voronoï 2D et 3D*, Thèse de l'Universtié Joseph Fourier, Grenoble I (1995)

[Blum1967] Harry Blum, *A transformation for extracting new descriptors of shape*, Communications of the ACM 18 (1967), pp 509-517

[Blum1973] Harry Blum, *Biological Shape and Visual Science*, Journal of Theoretical Biology 38 (1973), pp 205-287

[Breton2002] R. Breton, E. Andres, *Vectorisation d'une courbe discrète standard 2D*, AFIG (2002)

[Chakravarthy2003] V. Srinivasa Chakravarthy, Bhaskar Kompella , *The shape of handwritten characters*, Pattern Recognition Letters 24 (2003), pp 1901-1913

[Choi2003] Wai-Pak Choi, Kin-Man Lam, Wan-Chi Siu, *Extraction of the Euclidian skeleton based on a connectivity criterion*, Pattern Recognition 36 (2003), pp 721-729

[Cloppet2000] F. Cloppet, J. M. Oliva, G. Stamon, *Angular Bisector Network, a Simplified Generalized Voronoï Diagram : Application to Processing Complex Intersections in Biomedical Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1) (2000), pp 120-128

[Datta1997] - cité aussi page 458

Amitava Datta, S.K. Parui , *Skeletons from dot patterns : A neural network approach*, Pattern Recognition Letters 18 (1997), pp 335-342

[Freeman1970] H. Freeman, *Boundary encoding and processing*, B.S. Lipkin and A. Rosenfeld editors, Picture Processing and Psychopictorics, New-York Academic (1970), pp 241-266

[Gold1996] S. Gold, . Rangarajan, *A graduated assignment algorithm for graph matching*, IEEE Transactions on Pattern Analysis and Machine Intelligence 18(4) (1996), pp 377-288

[Huang2003] Lei Huang, Genxun Wan, Changpin Liu, *An Improved Parallel Thinning Algorithm*, Internation Conference on Document Analysis and Recognition (2003), pp 780-783

[Jang1992] B. K. Jang, R. T. Chin, *One-pass Parallel Thinning Analysis, Properties, and Quantitative evaluation*, IEEE Transactions on Pattern Analysis and Machine Intelligence 14(11) (1992), pp 869-885

[Kalmár1999] Zsolt Kalmár, Zsolt Marczell, Csaba Szepesvári, András Lörincz, *Parallel and robust skeletonization built on self-organizing elements*, Neural Networks 12 (1999), pp 163-173

[Kruskal1956] J. Kruskal, *On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem*, Proceeding American Mathematical Society 7(48) (1956)

[Lam1992] L. Lam, S.-W. Lee, C. Y. Suen, *Thinning Methodologies- A Comprehensive Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence 14(9) (1992), pp 869-885

[L'Homer2000] Erik L'Homer, *Extraction of strokes in handwritten characters*, Pattern Recognition 33 (2000), pp 1147-1160

[Liu2003] Zhi-Yong Liu, Kai-Chun Chiu, Lei Xu, *Strip line detection and thinning by RPCL-based local PCA*, Pattern Recognition Letters 24 (2003), pp 2335-2344

[Marthon1979] P. Marthon, A. Bruel, G. Biguet, *Squelettisation par calcul d'une fonction discriminante sur un voisinage de 8 points*, Actes du second congrès AFCET : Reconnaissance des Formes et Intelligence Articielles 107 (1979), pp 114-

[Ogniewicz1992] R. L. Ogniewicz, *Discrete Voronoi Skeletons*, PhD Thesis, Swiss Federal Institute of Technology (1992)

[Ogniewicz1995] R. L. Ogniewicz, O. Kubler, *Hierarchic Voronoi skeletons*, Pattern Recognition 28 (1995), pp 343-359

[Reveillès1991] J. P. Reveillès, *Géométrie discrète, calculs en nombre entiers et algorithmique*, Thèse, Université Louis Pasteur, Strasbourg (1991)

[Rosenfeld1986] A. Rosenfeld, *Axial representations of shape*, Computer Vision, Grahics, and Image Processing 33 (1986), pp 156-173

[Ruberto2004] - cité aussi page 459

Cecilia Di Ruberto, *Recognition of shapes by attibuted skeletal graphs*, Pattern Recognition 37 (2004), pp 21-31

[Singh2000] Rahul Singh, Vladimir Cherkassky, Nikolaos Papanikolopoulos, *Self-Organizing Maps for the Skeletonization of Sparse Shapes*, IEEE Transactions on Neural Networks 11(1) (2000)

[Su2003] Yih-Ming Su, Jhing-Fa Wang, *A novel stroke extraction method for Chinese characters using Gabor filter*, Pattern Recognition 36 (2003), pp 635-647

[Thiel1994] E. Thiel, *Les distances de chanfrein en analyse d'images : fondements et applications*, Thèse, Université Joseph Fourier, Grenoble I (1994)

[Vittone1999] J. Vittone, *Caractérisation et reconnaissance de droites et de plans en géométrie discrète*, Thèse, Université Joseph Fourier, Grenoble I (1999)

[Yeung1996] Daniel S. Yeung, H. S. Fong, *A fuzzy substroke extractor for handwritten chinese characters*, Pattern Recognition 29 (1996), pp 1963-1980

[ZhangY1997] You Yi Zhang, *Redundancy of parallel thinning*, Pattern Recognition Letters 18 (1997), pp 27-35

[ZhangB2004] Baibo Zhang, Changshui Zhang, Xing Yi, *Competitive EM algorithm for finite mixture models*, Pattern Recognition 37 (2004), pp 131-144

[Zhong1999] David X. Zhong, Hong Yan, *Pattern skeletonziation using run-length-wise processing for intersection distorsion problem*, Pattern Recognition Letters 20 (1999), pp 833-846

[7] Réseaux de neurones

[Bishop1995] - cité aussi page 453

C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press (1995), pp 0-

[Bottou1991] - cité aussi page 453

L. Bottou, *Une approche théorique de l'apprentissage connexionniste, Application à la reconnaissance de la parole*, Thèse de l'Université de Paris Sud, Centre d'Orsay (1991), pp 0-

[Broyden1967] C.G. Broyden, *Quasi-Newton methods and their application to function minimization*, Math. Comput 21 (1967), pp 368-

[Cottrel1995] - cité aussi page 453

M. Cottrel, B. Girard, M. Mangeas, C. Muller, *Neural modeling for time series : a statistical stepwise methode for weight elimination*, IEEE Transaction On Neural Networks 6(6) (1995), pp 0-

[Cybenko1989] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Controls, Signals, and Systems 2 (1989), pp 303-314

[Davidon1959] C. W. Davidon, *Variable metric method for minimization*, A.E.C. Research and Development Report, ANL-5990 (1959)

[Driancourt1996] X. Driancourt, *Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique, Application à la reconnaissance de la parole*, Thèse de l'Université de Paris Sud, Centre d'Orsay (1996)

[Fletcher1963] R. Fletcher, M. J. D. Powell, *A rapidly convergent descent method for minimization*, , Computer Journal (1963), pp 6-163 168

[Fletcher1993] R. Fletcher, *An overview of Unconstrained Optimization*, Numerical Analysis Report NA/149 (1993)

[Kullback1951] S. Kullback, R. A. Leibler , *On information and sufficiency*, Ann. Math. Stat. 22 (1951), pp 79-86

[LeCun1985] Y. Le Cun, *Une procédure d'apprentissage pour réseaux à seuil asymétrique*, Cognita (1985), pp 599-604

[Moré1977] J. J. Moré, *The Levenberg-Marquardt algorithm : Implementation and theory*, Proceedings of the 1977 Dundee Conference on Numerical Analysis, G. A. Watson, ed., Lecture Notes in Mathematics, vol. 630, Springer-Verlag, Berlin (1977), pp 105-116

[Rumelhart1986] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning internal representations by error propagation*, Parallel distributed processing : explorations in the microstructures of cohniyionn MIT Press, Cambridge (1986)

[Saporta1990] - cité aussi page 454

Gilbert Saporta , *Probabilités, analyse des données et statistique*, Editions Technip (1990)

[Song1997] Song Wang, Shaowei Xia, *Self-organizing algorithm of robust PCA based on single layer NN*, Proceedings of the 4th International Conference Document Analysis and Recognition (ICDAR) (1997)

[8] Support Vector Machines

[Boser1992] B. E. Boser, I. M. Guyon, V. Vapnik, *A training algorithm for optimal margin classifiers*, In Fifth annual Workshop on Computational Learning Theory, Pittsburg, ACM (1992)

[Burges1997] C. J. C. Burges, B. Schölkopf, *Improving the accuracy and speed of support vector machines*, In M. Mozer, M. Jordan and T. Petsche, editors, Advance in Neural Information Processing System 9 (1997), pp 375-381

[Burges1998] - cité aussi page 453

C. J. C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery 2(2) (1998), pp 955-974

[Cherkassky2004] Vladimir Cherkassky, Yunqian Ma, *Practical selection of SVM parameters and noise estimation for SVM regression*, Neural Networks 17 (2004), pp 113-126

[Mattera1999] D. Mattera, S. Haykin, *Support Vector Machine for dynamic reconstruction of a chaotic system*, Editions B. Schölkopf, J. Burges, A. Smola, Advances in Kernel Methods : Support Vector Machine, Cambridge, MA :MIT Press (1999)

[Müller2001] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Studa, Barhnard Schölkopf, *An Introduction to Kernel-Based Learning Algorithms*, IEEE Transactions on Neural Networks 12(2) (2001), pp 181-201

[Smola1998] Alex. J. Smola, Bernhard Schölkopf, *A Tutorial on Support Vector Regression*, NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK (1998)

[Smola2004] Alex. J. Smola, Bernhard Schölkopf, *A Tutorial on Support Vector Regression*, Statistics and Computing 14 (2004), pp 199-222

[Vapnik1979] - cité aussi page 455

V. Vapnik, *Estimation of Dependances Based on Empirical Data, (in Russian)*, Nauka, Moscow, traduction anglaise : Springer-Verlag, New-York, 1982 (1979), pp 0-

[Vapnik1995] V. Vapnik, *The nature of Statistical Learning Theory*, Springer-Verlag, New-York (1995)

[Vapnik1998] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, Inc., New-York (1998)

[Zhan2005] Yiqiang Zhan, Dinggang Shen, *Design efficient support vector machine for fast classification*, Pattern Recognition 38 (2005), pp 157-161

[9] Modèles de Markov cachés

[Baum1968] - cité aussi page 453

L. E. Baum, G. R. Sell, *Growth transformation for functions on manifolds*, Pac. J. Math. 27 (1968), pp 211-227

[Baum1972] - cité aussi page 453

L. E. Baum, *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process*, Inequalities 3 (1972), pp 1-8

[Bicego2003] - cité aussi page 453

Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *A sequential pruning strategy for the selection of the number of states in hidden Markov Models*, Pattern Recognition Letters 24 (2003), pp 1395-1407

[Bottou1991] - cité aussi page 453

L. Bottou, *Une approche théorique de l'apprentissage connexionniste, Application à la reconnaissance de la parole*, Thèse de l'Université de Paris Sud, Centre d'Orsay (1991), pp 0-

[Celeux1985] G. Celeux, J. Diebolt, *The SEM algorithm : a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem*, Computational Statistics Quarterly 2(1) (1985), pp 73-82

[Celeux1995] Gilles Celeux, Didier Chauveau, Jean Diebolt, *On stochastic version of the EM algorithm*, Rapport de recherche de l'INRIA nř2514 (1995)

[Chen1994] - cité aussi page 458

Mou-Yen Chen, Amlan Kundu, Jian Zhou, *Off-line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network*, IEEE Transations on Pattern Analysis and Machine Intelligence 16(5) (1994)

[Dempster1977] - cité aussi page 453

A. P. Dempster, N. M. Laird, D. B. Rubin, *Maximum-Likelihood from incomplete data via the EM algorithm*, Journal of Royal Statistical Society B 39 (1977)

[Dijkstra1971] - cité aussi page 456

E. W. Dijkstra, *Hierarchical ordering of sequential processes*, Acta Informatica 1,2 (1971)

[Levinson1983] - cité aussi page 454

S. E. Levinson, L. R. Rabiner, N. M. Sondhi, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, Bell System Technical Journal 52(4) (1983)

[Rabiner1986] - cité aussi page 454

L. R. Rabiner, B. H. Juang, *An introduction to hidden Markov models*, IEEE ASSP Magazine January (1986)

[10] Sélection d'architecture

[Augustin2001] - cité aussi page 453

E. Augustin, *Reconnaissance de mots manuscrits par systèmes hybrides réseaux de neurones et modèles de Markov cachés*, Thèse de l'Université Paris V (2001)

[Balasubramanian1993] - cité aussi page 453

V. Balasubramanian, *Equivalence and Reduction of Hidden Markov Models*, Massachussetts Institute of Technology Artificial Intelligence Laboratory, A.I. Technical Report No. 1370 (1993)

[Bicego2003] - cité aussi page 453

Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *A sequential pruning strategy for the selection of the number of states in hidden Markov Models*, Pattern Recognition Letters 24 (2003)

[Durand2003] - cité aussi page 458

Jean-Baptiste Durand, *Modèles à structure cachée : inférence, sélection de modèles et applications*, Thèse de l'Université Joseph Fourier (2003)

[Kamp1985] - cité aussi page 454

Y. Kamp, *State Reduction in Hidden Markov Chains Used for Speech Recognition*, IEEE Transactions of Acoustics, Speech and Signal Processing Vol. ASSP-33, No. 5 (1985)

[Ito1992] H. Ito, S.-I. Amari, K. Kobayashi, *Identifiability of hidden Markov information sources and their minimum degress of freedom*, IEEE Transaction on information theory 38(2) (1992), pp 324-333

[Levinson1983] - cité aussi page 454

S.E. Levinson, L. R. Rabiner, N. M. Sondhi, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, Bell System Technical Journal Vol 52. No 4 (1983), pp 257-285

[Pearson1972] E. S. Pearson, H. O. Hartley, *Biometrika Tables for statisticians*, Cambridge University Press (1969-1972)

[Saporta1990] - cité aussi page 454

Saporta Gilbert, *Probabilités, analyse des données et statistique*, Editions Technip (1990)

[Ziv1992] - cité aussi page 455

J. Ziv, N. Merhav, *Estimating the number of states of a finite state source*, IEEE Transactions on Information Theory 38(1) (1992)

[11] Graphes d'observations

[Bengio1992] - cité aussi page 453

Y. Bengio, R. De Mori, G. Flammia, *Global Optimization of a Neural Network - Hidden Markov Model Hybrid*, IEEE Transactions on Neural Network 3(3) (1992)

[Bengio1996] - cité aussi page 453

Yoshua Bengio, *Neural Networks for Speech and Sequence Recognition*, International Thomson Publishing Inc. (1996)

[Chen1994] - cité aussi page 458

Mou-Yen Chen, Amlan Kundu, Jian Zhou, *Off-line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network*, IEEE Pattern Analysis and Machine Intelligence 16(5) (1994)

[Dijkstra1971] - cité aussi page 456

E. W. Dijkstra, *Hierarchical ordering of sequential processes*, Acta Informatica 1,2 (1971)

[Lallican1999] - cité aussi page 454

P. M. Lallican, *Reconnaissance de l'écriture manuscrite hors-ligne : utilisation de la chronologie restaurée du tracé*, Thèse soutenue de l'Université de Nantes Ireste (1999)

[Mohri2000] Mehryar Mohri, *Minimization Algorithms for Sequential Transducers*, Theoretical Computer Science 234 (2000), pp 177-201

[Mohri2002a] Mehryar Mohri, Fernando C. N. Pereira, Michael Riley, *Weighted Finite-State Transducers in Speech Recognition*, Computer Speech and Language 16(1) (2002), pp 69-88

[Mohri2002b] Mehryar Mohri, *Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers*, International Journal of Foundations of Computer Science 13(1) (2002), pp 129-143

[Pereira1997] Fernando C. N. Pereira, Michael Riley, *Speech Recognition by Composition of Weighted Finite Automata*, In E. Roche and Y. Schabes, editors, Finite-State Language Processing. MIT Press, Cambridge, Massachusetts. (1997)

[Saon1997a] G. Saon, *Modèles markoviens uni- et bidimensionnels pour la reconnaissance de lŠécriture manuscrite hors-ligne*, Thèse de l'Université Henri Poincaré - Nancy I, Vandœuvre-lès-Nancy (1997)

[12] Classification non supervisée

[Balakrishnan1996] P. V. Sundar Balakrishnan, Martha Cooper, Varghese S. Jacob, Phillip A. Lewis, *Comparative performance of the FSCL neural net and K-means algorithm for market segmentation*, European Journal of Operation Research 93 (1996), pp 346-357

[Bandyopadhyay2004] Sanghamitra Bandyopadhyay, *An automatic shape independant clustering technique*, Pattern Recognition 37 (2004), pp 33-45

[Barandela2003] - cité aussi page 457

R. Barandela, J.S. Sanchez, V. Garcia, E. Rangel, *Strategies for learning in class imbalance problems*, Pattern Recognition 36 (2003), pp 849-851

[Biernacki2001] C. Biernacki, G. Deleux, G. Govaert, *Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood*, IEEE Transactions on Image Analysis and Machine Intelligence 22(7) (2001), pp 719-725

[Camastra2003] F. Camastra, A. Vinciarelli, *Combining Neural Gas and Learning Vector Quantization for Cursive Character Recognition*, Neurocomputing 51 (2003), pp 147-159

[Cheung2003] Yiu-Ming Cheung, *k^*-Means : A new generalized k-means clustering algorithm*, Pattern Recognition Letters 24 (2003), pp 2883-2893

[Celeux1985] - cité aussi page 465

G. Celeux, J. Diebolt, *The SEM algorithm : a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem*, Computational Statistics Quarterly 2(1) (1985), pp 73-82

[Celeux1995] - cité aussi page 465

Gilles Celeux, Didier Chauveau, Jean Diebolt, *On stochastic version of the EM algorithm*, Rapport de recherche de l'INRIA nř2514 (1995), pp 0-

[Davies1979] - cité aussi page 458

D. L. Davies, D. W. Bouldin, *A cluster Separation Measure*, IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 1(2) (1979), pp 0-

[Dempster1977] - cité aussi page 453

A. P. Dempster, N. M. Laird, D. B. Rubin, *Maximum-Likelihood from incomplete data via the EM algorithm*, Journal of Royal Statistical Society B 39 (1977), pp 1-38

[Figueiredo2002] M. A. T. Figueiredo, A. K. Jain, *Unsupervised learning of finite mixture models*, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(3) (2002), pp 381-396

[Goodman1954] L. Goodman, W. Kruskal, *Measures of associations for cross-validations*, J. Am. Stat. Assoc. 49 (1954), pp 732-764

[Herbin2001] - cité aussi page 459

M. Herbin, N. Bonnet, P. Vautrot, *Estimation of the number of clusters and influence zones*, Pattern Recognition Letters 22 (2001), pp 1557-1568

[Hoti2004] - cité aussi page 459

Fabian Hoti, Lasse Holmström, *A semiparametric density estimation approach to pattern classification*, Pattern Recognition 77 (2004), pp 409-419

[Kohonen1982] T. Kohonen, *Self-organized formation of topologically correct feature maps*, Biol. Cybern. 43 (1982), pp 59-69

[Kohonen1997] - cité aussi page 459

T. Kohonen, *Self-Organizing Map*, Springer (1997), pp 0-

[Kothari1999] Ravi Kothari, Dax Pitts, *On finding the number of clusters*, Pattern Recognition Letters 20 (1999), pp 405-416

[Liu2003] - cité aussi page 462

Zhi-Yong Liu, Kai-Chun Chiu, Lei Xu, *Strip line detection and thinning by RPCL-based local PCA*, Pattern Recognition Letters 24 (2003), pp 2335-2344

[Lo1991] Z. Lo, B. Bavarian, *On the rate of convergence in topology preserving neural networks*, Biological Cybernetics 63 (1991), pp 55-63

[Martinetz1993] T. Martinetz, S. Berkovitch, K. Schulten, *"Neural Gas" network for vector quantization and its application to time-series prediciton*, IEEE Trans. Neural Networks 4(4) (1993), pp 558-569

[Saporta1990] - cité aussi page 454

Gilbert Saporta, *Probabilités, analyse des données et statistique*, Editions Technip (1990), pp 0-

[Silverman1986] - cité aussi page 460

B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability, Chapman and Hall, London 26 (1986), pp 0-

[Waard1995] - cité aussi page 460

W. P. de Waard, *An optimised distance method for character recognition*, Pattern Recognition Letters 16 (1995), pp 499-506

[Wu2004] Sitao Wu, Tommy W. S. Chow, *Custering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density*, Pattern Recognition 37 (2004), pp 175-188

[Xu1993] L. Xu, A. Krzyzak, E. Oja, *Rival penalized competitive learning for clustering analysis, rbf net and curve detection*, IEEE Trans. Neural Networks 4 (1993), pp 636-649

[ZhangB2004] - cité aussi page 463

Baibo Zhang, Changshui Zhang, Xing Yi, *Competitive EM algorithm for finite mixture models*, Pattern Recognition 37 (2004), pp 131-144

[ZhangYG2004] Yun Gang Zhang, Chan Shui Zhang, David Zhang, *Distance metric learning by knwoledge embedding*, Pattern Recognition 37 (2004), pp 161-163

[13] Classification supervisée

[Chang1974] C. L. Chang, *Finding prototypes for nearest neighbor classifiers*, IEEE Transactions on Computer 23(11) (1974), pp 1179-1184

[Hart1968] P. E. Hart, *The Condensed nearest neighbor rule*, IEEE Transactions on Information Theory 14 (1968), pp 515-516

[Bezdek2001] J.C. Bezdek, L.I. Kuncheva, *Nearest prototype classifier designs : An experimental study*, International Journal of Intelligent Systems 16(12) (2001), pp 1445-1473

[Frasconi1997] P. Frasconi, M. Gori, G. Soda, *Links between LVQ and Backpropagation*, Pattern Recognition Letters 18 (1997), pp 303-310

[Kim2003] Sang-Woon Kim, B.J. Oommen, *Enhancing prototype reduction schemes with LVQ3-type*, Pattern Recognition 36 (2003), pp 1083-1093

[Kohonen1982] - cité aussi page 468

T. Kohonen, *Self-organized formation of topologically correct feature maps*, Biol. Cybernet. 43 (1982), pp 59-69

[Kohonen1995] T. Kohonen, *Self-organizing Map*, Springer, Berlin (1995)

[Linde1980] Y. Linde, A. Buzo, R. M. Gray, *An algorithm for vector quantizer design*, IEEE Transactions on Commun. 28 (1980), pp 84-95

[Vakil2003] Mohhamad-Taghi Vakil-Baghmisheh, Nikola Pavesic, *Premature clustering phenomenon and new training algorithms for LVQ*, Pattern Recognition 36 (2003), pp 1901-1912

[Vapnik1998] - cité aussi page 464

V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York (1998), pp 0-

[14] Distance d'édition

[Damerau1964] F. J. Damerau, *A technique for computer detection and correction of spelling errors*, Commun. ACM 7(3) (1964), pp 171-176

[Kripasundar1996] V. Kripasunder, G. Seni, R. K. Srihari, *Generating edit distance to incorporate domain information*, CEDAR/SUNY (1996)

[Levenstein1966] V. I. Levenstein, *Binary codes capables of correctiong deletions, insertions, and reversals*, Soviet Physics Doklady 10(8) (1966), pp 707-710

[Seni1996] Giovanni Seni, V. Kripasundar, Rohini K. Srihari, *Generalizing edit distance to incorporate domain information : handwritten text recognition as a case study*, Pattern Recognition 29 (1996), pp 405-414

[Waard1995] - cité aussi page 460

W. P. de Waard, *An optimised minimal edit distance for hand-written word recognition*, Pattern Recognition Letters 1995 (1995), pp 1091-1096

[Wagner1974] R. A. Wagner, M. Fisher, *The string-to-string correction problem*, Journal of the ACM 21 (1974), pp 168-178

[15] Espaces métriques

[Apostolico1985] A. Apostolico, *The Myriad virtues of subword trees*, Combinatorial Algorithms on Words, Springer-Verlag (1985), pp 85-96

[Arya1994] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, Angela Y. Wu, *An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions*, JACM 45(6) (1994), pp 891-923

[Beckmann1990] N. Beckmann, H. P. Kriegel, P. Schneider, B. Seeger, *The R^*-tree : an efficient and robust access method for points and rectangles*, Proceedings of SIGMOD conference, Atlantic City (1990), pp 322-331

[Berchtold1996] S. Berchtold, D. A. Keim, H. P. Kriegel, *The X-Tree : An index structure for high dimension data*, Proceedings of the 22nd Internation Conference on Very Large Databases, Bombay, India (1996)

[Bustos2001] B. Bustos, G. Navarro, E. Chavez, *Pivot Selection Techniques for Proximity Searching in Metric Spaces*, Proceedings of SCCC'01 (2001), pp 33-40

[Chavez1999] E. Chavez, G. Navarro, R. Baeza-Yates, J. Marroquin, *Searching in metric spaces*, Technical report TR/DCC-99-3, Departement of Computer Science, University of Chile (1999)

[D'Haes2003] Wim D'Haes, Dirk van Dyck, Xavier Rodet, *PCA-based branch and bound search algorithms for computing K nearest neighbours*, Pattern Recognition Letters 24 (2003), pp 1437-1451

[Dupré2003] - cité aussi page 461

X. Dupré, *Optimization of cursive words recognition and nearest neighbors search in metric spaces*, International Conference on Image and Signal Processing (ICISP), Agadir, Morroco (2003), pp 608-615

[Faragó1993] A. Faragó, T. Linder, G. Lugosi, *Fast Nearest-Neighbor Search in Dissimilarity Spaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence 15(9) (1993), pp 957-962

[Fukunaga1975] K. Fukunaga, P. M. Nevada, *A branch and bound algorithm for computing k nearest neighbors*, IEEE Transactions on Computers 24 (1975), pp 750-753

[Guttman1984] A. Guttman, *R-Trees : A Dynamic Index Structure for Spatial Searching*, Proceedings ACM SIGMOD (1984), pp 47-57

[Madhvanath2001] - cité aussi page 456

S. Madhvanath, V. Krpasundar, V. Govindaraju, *Syntactic methodology of pruning large lexicons in cursive script recognition*, Pattern Recognition 34 (2001), pp 37-46

[Micó1996] Luisa Micó, Jose Oncina, Rafeal C. Carrasco, *A fast branch and bound nearest neighbours classifier in metric space*, Pattern Recognition Letters 17 (1996), pp 731-739

[Moreno2003] Francisco Moreno-Seco, Luisa Mico, Jose Oncina, *A modification of the LAESA algorithm for approximated k-NN classification*, Pattern Recognition Letters 24 (2003), pp 47-53

[Levenstein1966] - cité aussi page 469

V. I. Levenstein, *Binary codes capables of correctiong deletions, insertions, and reversals*, Soviet Physics Doklady 10(8) (1966), pp 707-710

[Navarro2001] G. Navarro, *A Guided tour to approximate string matching*, ACM Computing Survey 33(1) (2001), pp 31-88

[Ramasubramanian2000] V. Ramasubramanian, Kuldip K. Paliwal, *Fast nearest-neighbor search algorithms based on approximation-elimination search*, Pattern Recognition 33 (2000), pp 1497-1510

[Reinert1979] M. Reinert, *Classification ascendante hiérarchique pour l'analyse de contenu et traitement statistique de corpus*, PhD Thesis, Université Pierre et Marie Curie, Paris VI (1979)

[Rico-Juan2003] J. R. Rico-Juan, L. Mico, *Comparison of AESA and LAESA search algorithms using string and tree-edit-distances*, Pattern Recognition Letters 24 (2003), pp 1417-1426

[Saporta1990] - cité aussi page 454

Gilbert Saporta, *Probabilités, analyse des données et statistique*, Editions Technip (1990), pp 0-

[Sellis1987] T. Sellis, N. Roussopoulos, C. Faloutos, *The R+tree - a Dynamic Index for Multi-Dimensional Objects*, Proceedings of the 13th VLDB conference (1987), pp 507-518

[Ulhmann1991] J. Ulhmann, *Implementing trees to satisfy general proximity / similarity queries with metrics trees*, Information Processing Letters 40 (1991), pp 175-179

[Wagner1974] - cité aussi page 469

R. A. Wagner, M. Fisher, *The string-to-string correction problem*, Journal of the ACM 21 (1974), pp 168-178

[Wu2002] Yingquan Wu, Krassimir Ianakiev, Venu Govindaraju, *Improved k-nearest neighbors classification*, Pattern Recognition 35 (2002), pp 2311-2318

[Yianilos1993] P. N. Yianilos, *Data structures and algorithms for nearest neighbor search in general metric spaces*, Proc. of the 4th Annual ACM-SIAM Symp. on Discrete Algorithms 311 (1993), pp 321-

[16] N-grammes

[Béchet2004] F. Béchet, R. De Mori, D. Janiszek, *Data augmentation and language model adaptation using singular value decomposition*, Pattern Recognition Letters 24 (2004), pp 15-19

[Bicego2003] - cité aussi page 453
Manuele Bicego, Vittorio Murino, Mario A.T. Figueiredo, *A sequential pruning strategy for the selection of the number of states in hidden Markov Models*, Pattern Recognition Letters 24 (2003), pp 1395-1407

[Govindaraju2002] Venu Govindaraju, Peter Slavík, Hanhong Xue, *Use of Lexicon Density in Evaluating Word Recognizers*, IEEE Pattern Analysis and Machine Intelligence 24(6) (2002), pp 789-800

[Perraud2003] Freddy Perraud, Christian Viard-Gaudin, Emmanuel Morin, Pierre-Michel Lallican, *N-Gram and N-Class Models for On line Handwriting Recognition*, Internation Conference on Document Analysis and Recognition (2003), pp 1053-1057

[Yamamoto2003] Hirofumi Yamamoto, Shuntara Isogai, Yoshinori Sagisaka, *Multi-class composite N-gram language model*, Speech Communication (2003)

[17] Receiving Operator Characteristic (ROC)

[Agarwal2005] **- jamais cité -** Shivani Agarwal, Thore Graepel, Ralf Herbich, Sariel Har-Peled, Dan Roth, *Generalization Bounds for the Area Under the ROC Curve*, Journal of Machine Learning Research 6 (2005), pp 393-425

[Saporta1990] - cité aussi page 454
Gilbert Saporta, *Probabilités, analyse des données et statistique*, Editions Technip (1990), pp 0-

[18] Analyse de documents

[Liu2000] Jinhui Liu, A.K. Jain, *Image-based form document retrieval*, Pattern Recognition 33 (2000), pp 503-513

[Yu1996] B. Yu, A.K. Jain, *A generic system for form dropout*, IEEE Transactions an Pattern Analysis and Machine Intelligence 18(11) (1996), pp 1127-1134

[Zheng2001] Y. Zheng, C. Liu, X. Ding, *Form frame line detection with directional single-connected chain*, Proc. Int'l Conf. Document Analysis and Recognition (2001), pp 699-703

[Zheng2006] Y. Zheng, *Handwriting Identification, Matching, and Indexing in Noisy Document Images*, Technical Report : LAMP-TR-129/CS-TR-4781/UMIACS-TR-2006-06, University of Maryland, College Park (2006)

année	**citations**	**année**	**citations**
toutes	**254**		
1951	1	1954	1
1956	1	1959	1
1961	1	1962	1
1963	1	1964	1
1965	1	1966	1
1967	2	1968	2
1970	2	1971	1
1972	1	1973	2
1974	3	1975	1
1977	3	1978	2
1979	4	1980	1
1982	2	1983	1
1984	2	1985	5
1986	5	1987	1
1989	2	1990	5
1991	6	1992	10
1993	6	1994	6
1995	16	1996	16
1997	15	1998	9
1999	11	2000	14
2001	12	2002	14
2003	36	2004	21
2005	3	2006	1

Table des matières

Index

B

C

E

G

H

I

N

Q

R

S

T

U

V

W

Z

gğ

Résumé

Ces travaux s'inscrivent dans le cadre de la reconnaissance de l'écriture manuscrite dite hors-ligne. Celle-ci consiste à déchiffrer des mots cursifs présents dans une image. Il n'existe pas encore de solution satisfaisante à ce problème si sa résolution est envisagée de telle manière qu'elle soit valable pour toute sorte de documents. Néanmoins, lorsque ceux-ci sont d'un type précis comme un chèque ou un formulaire, le contexte permet de façonner des contraintes ou des restrictions qui limitent la recherche et autorisent des solutions performantes. Il est plus facile de reconnaître un mot si celui-ci est censé être un nombre dans le cas d'un chèque ou un prénom dans le cas d'un formulaire. Plus généralement, le problème abordé par ces travaux est la reconnaissance de mots cursifs appartenant à un vocabulaire donné.

Le schéma principal des systèmes de reconnaissance de mots cursifs s'articule autour de deux étapes. Il est tout d'abord nécessaire de prétraiter l'image originale du document de manière à en extraire le mot à reconnaître puis à le segmenter en lettres ou morceaux de lettres appelés graphèmes. Ce résultat est ensuite traité à l'aide de modèles mathématiques qui analysent la forme de chaque graphème et leur séquençage. Cette étape probabiliste intègre le plus souvent des modèles de Markov cachés et constitue la reconnaissance proprement dite. Ces modèles sont en effet estimés à partir de grandes bases d'images dont la "connaissance" est résumée dans les coefficients du modèle.

L'étude bibliographique a montré que la recherche dans le domaine de la reconnaissance de l'écriture ne propose pas d'approche très éloignée du formalisme des modèles de Markov cachés. Cette étude s'est prolongée pour aboutir à une description détaillée du processus qui part de l'image pour arriver au résultat afin d'offrir un panorama de la recherche actuelle. Ces travaux ont débouché sur une suite de contributions parvenant à accroître les performances sans pour autant remettre en cause le processus général de la reconnaissance de l'écriture.

Ces améliorations, qu'on peut qualifier de locales, explorent trois thèmes. Elles ont concerné tout d'abord une meilleure prise en compte des erreurs inhérentes au traitement d'images dont la principale tâche se résume à la segmentation d'une image en graphèmes à partir d'heuristiques simples. La prise de décision a ensuite été accélérée afin de pouvoir accroître la complexité du problème de la reconnaissance tout en conservant des temps de traitement raisonnables - de l'ordre de quelques dixièmes de seconde -. Le traitement d'image en lui-même a été retouché afin d'étudier un possible remplacement des heuristiques par des apprentissages sur de grandes bases de données, par l'estimation d'une segmentation en graphèmes traitant les accents de manière séparée, puis lors de la restauration de caractères imparfaits.

L'ensemble de ces travaux est exposé dans un manuscrit qui essaye de proposer un inventaire exhaustif des méthodes utilisées lors de la reconnaissance de l'écriture manuscrite.

Mots-clé

reconnaissance, écriture manuscrite, modèles de Markov cachés, classification, traitement d'images, graphème