TD noté, jeudi 10 décembre 2009

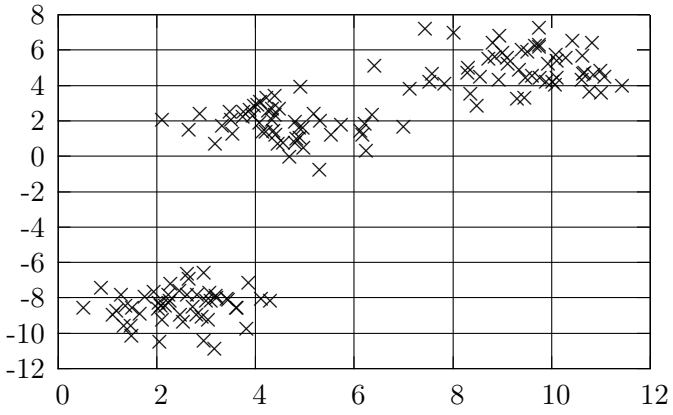
Le programme construit au fur et à mesure des questions devra être imprimé à la fin du TD et rendu au chargé de TD. Il ne faut pas oublier de mentionner son nom en commentaires au début du programme et l'ajouter sur chaque page. Les réponses autres que des parties de programme seront insérées sous forme de commentaires. Les définitions de fonctions proposées ne sont que des suggestions.

Glossaire :

barycentre — Un barycentre est le centre d'un ensemble de points $(x_i, y_i)_{1 \leqslant i \leqslant n}$. Dans le plan, il a deux coordonnées (X, Y) égales à $X = \frac{1}{n} \sum_i x_i$ et $Y = \frac{1}{n} \sum_i y_i$. Ses coordonnées le placent au milieu des points dans il est le barycentre : il est situé dans l'enveloppe convexe formée par l'ensemble des points.

centree — En dimension deux, même si c'est une expression employée dans la suite de l'énoncé, une loi normale de centre (x, y) n'est pas une expression correcte. On devrait dire une loi normale de moyenne (x, y). De même, cette loi n'a pas une variance $\sigma \in \mathbb{R}$, on devrait dire une variance $\begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix}$.

Les nuées dynamiques servent à construire automatiquement des classes dans un ensemble d'observations. C'est une façon de regrouper entre elles des observations qui sont proches les unes des autres. Prenons par exemple le nuage de points suivant qui inclut trois sous-nuages.



Le nuage de points contient trois sous-ensembles de points. Chacun est un ensemble de points simulés selon une loi normale de variance 1 et de moyenne identique à l'intérieur d'un sous-ensemble.

1) La fonction `gauss` du module `random` permet de générer un nombre selon une loi normale. Le premier objectif est de créer une fonction qui retourne un ensemble de points simulés selon une loi normale de variance v et de centre (x, y). (2 points)

```
def sous_nuage (nb, x, y, v) :  # retourne une liste de 2-uples
```

2) On cherche à créer un nuage regroupant n sous-nuages de même variance 1 avec la fonction précédente. Chaque sous-nuage est centré autour d'une moyenne choisie aléatoirement selon une loi de votre choix. La fonction dépend de deux paramètres : le nombre de points dans chaque classe et le nombre de classes. (2 points)

```
def n_sous_nuages (n, nb) :     # retourne une liste de 2-uples
```

3) Dessiner le nuage avec le module `matplotlib` pour vérifier que le résultat correspond à vos attentes. On pourra s'appuyer sur l'extrait qui suit. (1 point)

```
import matplotlib.pyplot as plt
x   = [ ... ]
y   = [ ... ]
fig = plt.figure()
ax  = fig.add_subplot(111)
ax.plot (x,y, 'o')
plt.savefig ("im1.png") # ou plt.show () pour afficher le graphe
```

4) L'algorithme des nuées dynamiques commence par affecter chaque point à une classe choisie au hasard. Pour cette tâche, on pourra utiliser la fonction `randint` du module `random`. On veut créer une fonction qui retourne une classe aléatoire pour chaque point du nuage. Elle doit prendre comme entrée le nombre de classes souhaité. (2 points)

```
def random_class (points, n) :  # retourne une liste d'entiers
```

5) L'algorithme des nuées dynamiques répète ensuite alternativement deux étapes :

Etape 1 On calcule le barycentre de chaque classe.
Etape 2 On associe à chaque point la classe dont le barycentre est le plus proche (au sens de la distance euclidienne).

On propose de commencer par écrire une fonction qui retourne pour un point donné le barycentre le plus proche. (2 points)

```
def proche_barycentre (point, barycentres) :   # retourne un entier
```

6) La fonction suivante retourne le barycentre le plus proche pour chaque point. (2 points)

```
def association_barycentre (points, barycentres) :  # retourne une liste d'entiers
```

7) On découpe la première étape de la même façon :

1. Première fonction : calcule le barycentre d'une classe.
2. Seconde fonction : calcule le barycentre de toutes les classes.

Il faut implémenter ces deux fonctions. (3 points sans utiliser `numpy`, 4 points avec `numpy` et une fonction).

```
def barycentre_classe (points, classes, numero_class) :   # retourne un 2-uple
def tous_barycentres  (points, classes) :       # retourne une liste de 2-uples
```

8) L'algorithme commence par la création des classes (fonction `n_sous_nuages`) et l'attribution d'une classe au hasard (fonction `random_class`). Il faut ensuite répéter les fonctions `tous_barycentres` et `association_barycentre`. L'enchaînement de ces opérations est effectué par la fonction `nuees_dynamiques`. (2 points)

```
def nuees_dynamiques (points, nombre_classes) : # retourne une liste d'entiers
```

9) Dessiner le résultat permettra de vérifier que tout s'est bien passé, toujours avec un code similaire à celui-ci. (2 points)

```
import matplotlib.pyplot as plt
x1  = [ ... ]
y1  = [ ... ]
x2  = [ ... ]
y2  = [ ... ]
fig = plt.figure()
ax  = fig.add_subplot(111)
ax.plot (x1,y1, 'o')
ax.plot (x2,y2, 'x')     # ligne ajoutée, 'x', 'bo', ...
plt.savefig ("im2.png")  # 'rx', 'go', 'gs', 'bs', ...
```

10) Question facultative : comment savoir quand s'arrêter ? (0 point)