ENSAE TD noté, mardi 12 décembre 2017

Le programme devra être envoyé par mail au chargé de TD et au professeur. Toutes les questions valent 2 points.

1

1) Générer un ensemble aléatoire de 1000 nombres (X_i, Y_i) qui vérifie :

— X_i suit une loi uniforme sur $[0, 16]$

— $Y_i = \mathbb{1}_{\{[\sqrt{X_i}] \mod 2 = 0\}}$ où $[A]$ est la partie entière de A.

On pourra se servir de la fonction `random` du module `random`. Vous pourrez vérifier que le nuage de points correspond à ce qui est demandé en exécutant le code suivant (si vous êtes dans un notebook, n'oubliez pas `%matplotlib inline`).

```
import matplotlib.pyplot as plt
plt.plot(X, Y, '.')
```

2) Trier les points selon les X. L'instruction `list(zip(X,Y))` devrait vous mettre sur la piste.

3) On suppose que les Y sont triés selon les X_i croissants. Calculer la somme des différences entre les Y_i et la moyenne m des Y_i (en valeur absolue) sur un intervalle $[i, j]$, j exclu. Ecrire une fonction `def somme_diff(nuage, i, j)` qui exécute ce calcul qui correspond à $\sum_{k=i}^{j-1} |Y_k - m|$ avec $m = (\sum_{k=i}^{j-1} Y_k)/(j - i)$.

4) Soit i, j deux entiers, on coupe l'intervalle en deux : i, k et k, j. On calcule `somme_diff` sur ces deux intervalles, on compare leur somme à celle obtenue sur l'ensemble de l'intervalle. On écrit la fonction `def difference(nuage, i, j, k) :`. Elle doit calculer $|\Delta_i^k + \Delta_k^j - \Delta_i^j|$ où Δ correspond à la fonction `somme_diff`.

5) Le langage Python permet de passer une fonction à une autre fonction en tant qu'argument :

```
def fct(x, y):
    return abs(x-y)
def distance_list(list_x, list_y, f):
    return sum(f(x,y) for x,y in zip(list_x, list_y))
distance_list([0, 1], [0, 2], fct)
```

On veut réécrire les fonctions `def somme_diff(nuage, i, j, fct)` et `def difference(nuage, i, j, k, fct) :` de telle sorte que `somme_diff` calcule $(\sum_{k=i}^{j-1} \mathbf{fct}(Y_k, m)$.

6) On veut déterminer le k optimal, celui qui maximise la quantité précédente dans l'intervalle $[i, j] = [0, 16]$. On souhaite garder la fonction `fct` comme argument. Pour cela, implémenter la fonction `def optimise(nuage, i, j, fct) :`. Elle retourne le point de coupure et la quantité optimale.

7) Recommencer sur les deux intervalles trouvés $[i, k]$, $[k, j]$ et calculer le résultat.

8) Pouvez-vous imaginer une fonction récursive qui produit toutes les séparations. Entre deux séparations, tous les Y sont constants. Ecrire la fonction `def recursive(nuage, i, j, fct) :`.

9) Quel est le coût de la fonction `optimise` en fonction de la taille de l'intervalle ? Peut-on mieux faire (ce qu'on n'implémentera pas).

10) Comment l'algorithme se comporte-t-il lorsque tous les points sont distincts ?

ENSAE TD noté, mardi 12 décembre 2017

Le programme devra être envoyé par mail au chargé de TD et au professeur. Toutes les questions valent 2 points.

2

1) Générer un ensemble aléatoire de 1000 nombres (X_i, Y_i) qui vérifie :

- X_i suit une loi uniforme sur $[0, 16]$
- $Y_i = \sqrt{X_i}[\sqrt{X_i}]$ où $[A]$ est la partie entière de A.

On pourra se servir de la fonction `random` du module `random`. Vous pourrez vérifier que le nuage de points correspond à ce qui est demandé en exécutant le code suivant.

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(X, Y, '.')
```

2) Trier les points selon les X. L'instruction `list(zip(X,Y))` devrait vous mettre sur la piste.

3) On suppose que les Y sont triés selon les X_i croissants. Calculer la somme des différences au carré entre les Y_i et la moyenne m des Y_i sur un intervalle $[i, j]$, j exclu. Ecrire une fonction `def somme_diff(nuage, i, j)` qui exécute ce calcul qui correspond à $\sum_{k=i}^{j-1} (Y_k - m)^2$ avec $m = (\sum_{k=i}^{j-1} Y_k)/(j - i)$.

4) Soit i, j deux entiers, on coupe l'intervalle en deux : i, k et k, j. On calcule `somme_diff` sur ces deux intervalles, on compare leur somme à celle obtenue sur l'ensemble de l'intervalle. On écrit la fonction `def difference(nuage, i, j, k) :`. Elle doit calculer $|\Delta_i^k + \Delta_k^j - \Delta_i^j|$ où Δ correspond à la fonction `somme_diff`.

5) Le langage Python permet de passer une fonction à une autre fonction en tant qu'argument :

```
def fct(x, y): return (x-y)**2
def distance_list(list_x, list_y, f): return sum(f(x,y) for x,y in zip(list_x, list_y))
distance_list([0, 1], [0, 2], fct)
```

On veut réécrire les fonctions `def somme_diff(nuage, i, j, fct)` et `def difference(nuage, i, j, k, fct) :` de telle sorte que `somme_diff` calcule $(\sum_{k=i}^{j-1} \mathbf{fct}(Y_k, m)$.

6) On veut déterminer le k optimal, celui qui maximise la quantité précédente dans l'intervalle $[i, j] = [0, 16]$. On souhaite garder la fonction `fct` comme argument. Pour cela, implémenter la fonction `def optimise(nuage, i, j, fct) :`. Elle retourne le point de coupure et la quantité optimale.

7) Recommencer sur les deux intervalles trouvés $[i, k]$, $[k, j]$ et calculer le résultat.

8) Pouvez-vous imaginer une fonction récursive qui produit toutes les séparations. Entre deux séparations, tous les Y sont constants. Ecrire la fonction `def recursive(nuage, i, j, fct) :`.

9) L'algorithme produit beaucoup de points de coupures. On souhaite arrêter la récursion plus tôt en mettant un seuil sur la quantité obtenue $|\Delta_i^k + \Delta_k^j - \Delta_i^j|$ qui doit être supérieur à 50.

10) Quel est le coût de la fonction `optimise` en fonction de la taille de l'intervalle ? Peut-on mieux faire (ce qu'on n'implémentera pas).