ENSAE TD noté, mardi 23 octobre 2018

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

1

Lisez d'abord les deux dernières questions avant de commencer, elles pourraient vous orienter sur la meilleure façon d'écrire les résultats aux questions qui précèdent pour y répondre rapidement.

1) Générer un ensemble de $N = 1000$ couples aléatoires (X_i, Y_i) qui vérifient :
- X_i suit une loi normale de variance 1.
- $Y_i = 2X_i + \epsilon_i$ où ϵ_i suit une loi normale de variance 1.

Fonction suggérée : `def random_mat(N)` : Quelques fonctions utiles : `numpy.random.normal`.

2) On définit la matrice $M \in \mathbb{M}_{N,2}(\mathbb{R})$ définie par les deux vecteurs colonnes (X_i) et (Y_i). Choisir aléatoirement 20 valeurs dans cette matrice et les remplacer par `numpy.nan`. On obtient la matrice M_1. Quelques fonctions utiles : `numpy.random.normal`. Fonction suggérée : `def build_m1(mat,n = 20)` :.

3) Calculer $\mathbb{E}(X) = \frac{1}{N}\sum_i^N X_i$ et $\mathbb{E}(Y) = \frac{1}{N}\sum_i^N Y_i$. On ne tient pas compte des valeurs manquantes (donc il y aura moins de N valeurs à sommer). Quelques fonctions utiles : `numpy.isnan`.

Fonction suggérée : `def mean_no_nan(mat)` :

4) Remplacer les valeurs manquantes de la matrice M_1 par la moyenne de leur colonnes respectives. On obtient la matrice M_2.

5) On suppose qu'on dispose d'une fonction qui trie la matrice M_1 selon une colonne. Celle-ci remplace les valeurs manquantes en faisant la moyenne des valeurs qui l'entourent pour une colonne en particulier. Ci-dessous, un exemple pour la second colonne qui contient les y. Il faut appliquer cette fonction à chaque colonne.

x_1	y_1	
x_2	$\frac{2}{3}y_1 + \frac{1}{3}y_4$	(valeur manquante remplacée)
x_3	$\frac{1}{3}y_1 + \frac{2}{3}y_4$	(valeur manquante remplacée)
x_4	y_4	

6) On a deux méthodes pour compléter les valeurs manquantes, quelle est la meilleure ? Il faut vérifier numériquement en comparant $\|M - M_2\|^2$ et $\|M - M_3\|^2$. L'erreur la plus faible détermine la méthode la plus efficace. **Les réponses numériques doivent apparaître sur votre copie.**

7) Une experience réussie ne veut pas dire que cela fonctionne tout le temps. Recommencer 10 fois en changeant le nuage de points et les valeurs manquantes ajoutées.

Fonction suggérée : `def repetition(N = 1000,n = 20,nb = 10)` :

8) Et si on augmente le nombre de valeurs manquantes, donc de 100 à 1000 tous les 100, l'écart se creuse-t-il ou se réduit -il ? Montrez-le numériquement.

9) S'il n'y qu'une valeur manquante, peut-on sans changer le résultat (celui de la question 5) se passer de tri pour avoir un coût linéaire ?

10) Pour cette question, vous avez le choix entre implémenter la solution que vous proposez à la question précédente (choix sans doute le plus risqué) ou proposer une façon d'étendre la méthode dans le cas où il y a 3 dimensions. Pour remplacer les valeurs manquantes d'une colonne, on trie selon l'autre colonne. Maintenant qu'il y en a trois, laquelle choisir ?

ENSAE TD noté, mardi 23 octobre 2018

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

2

Lisez d'abord les deux dernières questions avant de commencer, elles pourraient vous orienter sur la meilleure façon d'écrire les résultats aux questions qui précèdent pour y répondre rapidement.

1) Générer un ensemble de $N = 1000$ couples aléatoires (X_i, Y_i) qui vérifient :
- X_i suit une loi normale de variance 1.
- $Y_i = 2X_i + \epsilon_i$ où ϵ_i suit une loi normale de variance 1.

Fonction suggérée : `def random_mat(N)` : Quelques fonctions utiles : `numpy.random.normal`.

2) On définit la matrice $M \in \mathbb{M}_{N,2}(\mathbb{R})$ définie par les deux vecteurs colonnes (X_i) et (Y_i). Choisir aléatoirement 20 valeurs dans cette matrice et les remplacer par `numpy.nan`. On obtient la matrice M_1. Fonction suggérée : `def build_m1(mat,n = 20)` :.

3) Calculer $\mathbb{E}(X) = \frac{1}{N}\sum_i^N X_i$ et $\mathbb{E}(Y) = \frac{1}{N}\sum_i^N Y_i$. On ne tient pas compte des valeurs manquantes (donc il y aura moins de N valeurs à sommer). Quelques fonctions utiles : `numpy.isnan`.

Fonction suggérée : `def mean_no_nan(mat)` :

4) Remplacer les valeurs manquantes de la matrice M_1 par la moyenne de leur colonnes respectives. On obtient la matrice M_2.

5) On considère le point de coordonnées (x, y), écrire une fonction qui retourne le point de la matrice M dont l'abscisse est la plus proche de x et pour lequel y n'est pas `numpy.nan`.

6) Pour chaque y manquant, on utilise la fonction précédente pour retourner le point dont l'abscisse et la plus proche et on remplace l'ordonnée y par celle du point trouvé. On fait de même avec les x manquant. Si les deux manquent, on remplacera par la moyenne. On construit la matrice ainsi M_3 à partir de M_1.

7) On a deux méthodes pour compléter les valeurs manquantes, quelle est la meilleure ? Il faut vérifier numériquement en comparant $\|M - M_2\|^2$ et $\|M - M_3\|^2$. **Les réponses numériques doivent apparaître sur votre copie.**

8) Une experience réussie ne veut pas dire que cela fonctionne. Recommencer 10 fois en changeant le nuages de points et les valeurs manquantes ajoutées.

Fonction suggérée : `def repetition(N = 1000,n = 20,nb = 10)` :

9) Et si on augmente le nombre de valeurs manquantes, l'écart se creuse-t-il ou se réduit -il ? Montrez-le numériquement.

10) Votre fonction de la question 5 a probablement un coût plus que linéaire. Il est probablement possible de faire mieux, si oui, il faut préciser comment et ce que cela implique sur les données. Il ne faut pas l'implémenter.