Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

MASTER 1

Programmation

Analyse de données - Scilab

Dans ce TP, on souhaite étudier des méthodes permettant d'apprendre les paramètres de modèles à partir d'observations qui suivent ce modèle. Les deux premiers exercices concernent des méthodes de régression linéaire, et le troisième présente une méthode de *clustering* (groupement) de données.

Exercice 1 - Moindres carrés.
On étudie dans cet exercice un problème de régression linéaire. On observe plusieurs couples (x_i, y_i) dont on suppose qu'ils sont les réalisations de variables aléatoires X_i et Y_i liées par une relation linéaire

$$Y_i = aX_i + b + \varepsilon_i, \tag{1}$$

où a et b sont des constantes et où les ε_i sont des variables de perturbation des données (ces variables sont supposées prendre des valeurs petites devant les Y_i). On cherche à partir des observations à estimer les paramètres a et b. L'estimateur le plus simple pour ce problème est l'estimateur des moindres carrés. C'est la solution du problème de minimisation

$$\arg\min_{a,b} \sum_{i=1}^{n} |ax_i + b - y_i|^2. \tag{2}$$

Le but de l'exercice est de créer une fonction qui prend en entrée tous les x_i et tous les y_i, et donne la solution (a, b) du problème (2). On commence en créant des données qui vont nous servir pour cet exercice et le suivant. Les lignes ci-dessous créent deux vecteurs x et y liés par la relation (1) avec $a = 3$ et $b = 1$.

```
N = 1000;
x = 10*rand(N,1,'uniform');
e = rand(N,1,'normal');
y = (3*x+1)+e;
```

On peut ensuite afficher le nuage de points correspondant grâce à l'une ou l'autre des commandes

```
plot(x,y,'+');
plot2d(x,y,style=-1);
```

Si l'on veut remplacer une partie des observations par des données aberrantes (c'est-à-dire des couples (x_i, y_i) qui ne sont pas liés par la relation (1), ces couples peuvent provenir d'erreurs de mesure par exemple), on peut ajouter les lignes suivantes :

```
p = 1/3;              // proportion de données aberrantes
y(N-floor(N*p):N) = 30*rand(floor(N*p)+1,1,'uniform');
```

La proportion de données aberrantes est contrôlée par le nombre p qui doit être choisi entre 0 et 1.

1. Expliquer ce que font précisément les différentes lignes de codes ci-dessus.

2. Dériver la somme (2) par rapport à a et b pour obtenir la solution du problème des moindres carrés.
3. Ecrire une fonction $[a, b] =$ `moindrescarres`(x, y) qui prend en entrée deux vecteurs de même taille x et y et calcule la solution du problème des moindres carrés (2).
4. Utiliser la fonction `moindrescarres` sur les données créées en début d'exercice (sans données aberrantes). Tracer sur un même graphique les points (x_i, y_i) et la droite de régression $v = at+b$ obtenue.

```
t = linspace(min(x),max(x),200);
v = b + a*t;
plot(t,v,'g');   // Trace la courbe en vert
```

5. Refaire la question précédente sur des observations comportant des données aberrantes. Que se passe t'il ?

Exercice 2 - Ransac.

L'algorithme des moindres carrés est peu robuste à la présence de données aberrantes (des couples (x_k, y_k) qui ne sont pas liés par la relation (1)). On étudie ici l'algorithme RANSAC (pour RANdom SAmple Consensus), un algorithme stochastique permettant d'estimer les paramètres d'un modèle mathématique à partir de données observées. Cet algorithme est fait pour distinguer parmi les données celles qui suivent le modèle paramétrique (ici l'équation (1)), que l'on appelle *inliers*, et celles qui ne le suivent pas (on les appelle *outliers*).

Le principe de RANSAC est le suivant. On commence par tirer au hasard deux couples de points parmi les données observées. On calcule ensuite les paramètres (a, b) de la droite passant par ces deux couples de points. On note C l'ensemble des couples (x_k, y_k) pour lesquels la distance $|ax_k + b - y_k|$ est plus petite qu'un seuil α fixé. Cet ensemble C est appelé consensus. On itère cette opération L fois, et le consensus C le plus grand trouvé au cours des itérations est conservé et validé si son cardinal est suffisamment grand. L'ingéniosité de RANSAC réside dans son caractère stochastique, qui lui évite de tester toutes les transformations possibles.

1. Ecrire une fonction $[a, b] =$ `droite`(s, t, s', t') qui calcule les paramètres de la droite $y = ax+b$ passant par les points (s, t) et (s', t').
2. Ecrire une fonction $[a, b, C] =$ `ransac`(x, y) qui prend en entrée deux vecteurs x et y de même taille et calcule par RANSAC les paramètres du modèle linéaire entre x et y. On pourra s'inspirer du pseudo-code de Ransac fourni ci-dessous. Prendre par exemple un nombre d'itérations M de l'ordre de 10^3 et un seuil $\alpha = 1$.
3. On reprend les données x et y créees à l'exercice précédent (avec données aberrantes). Tracer sur un même graphique les points (x_k, y_k), la droite de régression $v = at+b$ trouvée par Ransac, et la droite trouvée précédemment par moindres carrés (on pourra utiliser la commande `plot` avec différentes options de couleur). Commentez la différence entre la droite trouvée et la droite calculée par moindres carrés.
4. L'algorithme Ransac est stochastique : comme on tire aléatoirement des couples de points à chaque itération, le résultat final de Ransac n'est pas forcément le même si on le lance plusieurs fois sur les mêmes données. Testez la stabilité de l'algorithme pour plusieurs valeurs de M, du seuil α, et en fonction de la proportion p de données aberrantes.

Pseudo-code de Ransac

Entrée : vecteur x et y de longueur n, seuil α, nombre d'itérations M, seuil P.
Sortie : paramètres a et b du modèle.
$\beta = 0$
for $k = 1$ **to** M **do**
Tirer aléatoirement deux entiers l et j entre 1 et n (loi uniforme).

$[a_k, b_k] =$ `droite`(x_l, y_l, x_j, y_j)
Trouver l'ensemble C_k des couples (x_i, y_i) tels que $|a_k x_i + b_k - y_i| \leq \alpha$.
Si $\#C_k > \max(P, \beta)$ alors $C = C_k$, $\beta = \#C$, $a = a_k$ et $b = b_k$
Si $\beta > P$, retourner C, a et b.

Fin du pseudo-code

Exercice 3 ***Clustering*** **et K-moyennes.** Le but de cet exercice est d'implémenter une méthode de groupements de points (ou *clustering* en anglais) appelée K-moyennes. On suppose qu'on observe n points $P_1, \ldots, P_n$ du plan. Chaque point P_i a des coordonnées que l'on note (x_i, y_i) dans le plan. On veut diviser cet ensemble de points en K groupes disjoints $G_1, \ldots, G_K$, afin que chaque point appartienne au groupe dont la moyenne est la plus proche de lui. Plus précisément, on souhaite partitionner l'ensemble $\{P_1, \ldots, P_n\}$ en K groupes de points $G_1, \ldots, G_K$ de manière à minimiser la quantité

$$\sum_{k=1}^{K} \sum_{P_i \in G_k} \|P_i - \mu_k\|^2, \tag{3}$$

où μ_k est la moyenne du k-ième groupe de points.
Ce problème est difficile à résoudre numériquement, mais on peut le résoudre de manière approchée grâce à l'algorithme des K-moyennes (ou K-*means* en anglais). L'algorithme fonctionne comme suit.
On commence par initialiser les moyennes $\mu_1, \ldots, \mu_K$. On peut par exemple initialiser chaque μ_k comme un point aléatoire dans le domaine où vivent les points P_i. L'algorithme fonctionne ensuite en itérant plusieurs fois les deux étapes suivantes :

(a) **Mise à jour des groupes**. Mettre chaque observation P_i dans le groupe G_k dont la moyenne μ_k est la plus proche de P_i.

$$Groupe(P_i) = G_k \quad \text{avec} \quad k = \arg \min_{j=1,\ldots,K} \|P_i - \mu_j\|_2 \tag{4}$$

(b) **Mise à jour des moyennes**. Calculer les nouvelles moyennes $\mu_1, \ldots, \mu_K$ des groupes $G_1, \ldots, G_K$ créés à l'étape 1

$$\mu_k = \frac{1}{\#G_k} \sum_{P_i \in G_k} P_i. \tag{5}$$

On considère que l'algorithme a convergé lorsque les groupes G_k ne changent plus au cours des itérations.
On commence l'exercice en créant des données synthétiques :

```
clear;
u = ones(500,1)*[1,5]+rand(500,2,'normal');
v = ones(500,1)*[4,1]+rand(500,2,'normal');
w = ones(500,1)*[4,7]+rand(500,2,'normal');
P=[u;v;w];
plot(P(:,1),P(:,2),style=-1);
```

1. Expliquer précisément ce que font les lignes de code ci-dessus.
2. Ecrire une fonction $[G] =$ `kmeans`(P, K) qui implémente l'algorithme des K-moyennes. La fonction prend en entrée un nuage de points P du plan (sous la forme d'une matrice $n \times 2$) et un entier K. Elle calcule un vecteur G, de même longueur que P, tel que $G(i)$ est l'index du groupe auquel appartient le point P_i après convergence des K-moyennes.
3. Appliquer cet algorithme aux données créées ci-dessus avec $K = 3$. Afficher le résultat du *clustering*. On pourra utiliser un symbole différent pour les points des différents groupes trouvés, par exemple grâce à la commande :

```
for k=1:3
plot2d(P(find(G==k),1),P(find(G==k),2),-k);
end
```