TD n° 3: Méthode des moindres carrés

1 Introduction

Une méthode classique pour traiter des données expérimentales est la *régression linéaire*, qui consiste à chercher une droite qui passe «au mieux» à travers le nuage de points représentant les mesures. Les coefficients de cette droite sont en général donnés par la méthode des *moindres carrés*.

En supposant que le phénomène étudié suit un modèle linéaire $y = ax + b$ et que l'on dispose d'un ensemble de N données (x_i, y_i), on cherche les coefficients a et b qui minimisent l'erreur quadratique :

$$\chi(a,b) = \frac{1}{N}\sum_{i=1}^{N} |y_i - (ax_i + b)|^2. \tag{1}$$

La théorie des probabilités permet de montrer que le critère des moindres carrés correspond au *maximum de vraissemblance* pour un modèle linéaire bruité par des variables gaussiennes centrées, identiques et indépendantes. Un extremum de la fonction d'erreur χ est obtenu pour le couple $(\hat{a}, \hat{b})$ qui annule les deux dérivées partielles :

$$\left(\frac{\partial \chi}{\partial a}\right)_{a=\hat{a},b=\hat{b}} = 0, \qquad \left(\frac{\partial \chi}{\partial b}\right)_{a=\hat{a},b=\hat{b}} = 0. \tag{2}$$

On admettra que celui-ci est le minimum cherché.

2 Régression linéaire

2.1 Formulation

1. Écrire le système satisfait par les coefficients $\hat{a}$ et $\hat{b}$ minimisant la fonction erreur. On introduira les quantités :

$$S_x = \sum_{i=1}^{N} x_i \quad , \quad S_y = \sum_{i=1}^{N} y_i$$
$$S_{xx} = \sum_{i=1}^{N} x_i^2 \quad , \quad S_{xy} = \sum_{i=1}^{N} x_i y_i$$
$$S_{yy} = \sum_{i=1}^{N} y_i^2$$

2. Résoudre le système et exprimer $\hat{a}$ et $\hat{b}$ en fonction de $N, S_x, S_y, S_{xx}, S_{xy}$.

La qualité d'une régression linéaire est donnée par le *coefficient de corrélation* :

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2 \sum_{i=1}^{N}(y_i - \overline{y})^2}},$$

où

$$\overline{x} = \frac{S_x}{N},\ \overline{y} = \frac{S_y}{N}$$

sont les valeurs moyennes de x_i et y_i. De manière générale $|\rho| \leq 1$. Le cas $\rho = 1$ correspond à une approximation (*fit* en anglais) linéaire parfaite (i.e. les points des données sont alignés). Un bon "fit" correspond à une valeur proche de 1. Exprimer ρ en fonction des différentes sommes S.

2.2 Programmation

Rappels

— Pour chaque nouveau TD, utiliser la commande `mkdir` pour créer un nouveau répertoire de travail et en faire le répertoire courant en utilisant la commande `cd`.

— Pour tous vos fichiers sources (dont le nom doit se terminer par `.py`) , il est d'usage d'indiquer en commentaire dans la première ligne vos noms et prénoms, la date et l'objet du programme.

1. On se propose d'écrire un premier programme de régression linéaire qui calcule les coefficients $\hat{a}, \hat{b}$ et ρ de la régression linéaire à partir de données définies en début du programme. On pourra nommer ce script `regression_1.py`. Les deux lignes déclarant les tableaux `numpy` `tab_x` et `tab_y` et définissant les données sont les suivantes :
```
tab_x = np.array([1.0, 2.0, 4.0, 10.0])
tab_y = np.array([3.1, 5.2, 10.65, 25.77])
```
On calculera les sommes `Sx, Sy, Sxy, Sxx` et `Syy` directement après la définition des tableaux (et non dans des fonctions *ad hoc*). On définira alors trois fonctions :
— `def pente(N, Sx, Sy, Sxy, Sxx)`
— `def ordo_origine(N, Sx, Sy, Sxy, Sxx)`
— `def coeff_corr(N, Sx, Sy, Sxy, Sxx, Syy)`
qui calculent respectivement $\hat{a}, \hat{b}$ et ρ.
On appellera ces fonctions dans le programme principal afin de faire afficher sur le terminal de commande les résultats.

2. Puis, dans un second programme nommé `regression_2.py`, on effectuera la régression linéaire à partir de données définies dans un fichier de données nommé `points.dat`, le format des données est le suivant : cent couples (x_i, y_i) de réels de type `float` au format texte en 2 colonnes, un retour à la ligne séparant chaque couple.

3. On représentera graphiquement les données du fichier `points.dat` ainsi que la droite issue du calcul de régression linéaire pour cela on utilisera le module `pylab` de la librairie `matplotlib` (voir polycopié de cours) importé via la commande :
`import matplotlib.pylab as plt`.

3 Régression polynomiale d'ordre 2

3.1 Formulation

On suppose que le phénomène étudié suit un modèle quadratique $y = ax^2 + bx + c$ et que l'on dispose d'un ensemble de N données (x_i, y_i) $(i = 1, \cdots, N)$, on cherche les coefficients a, b

et c qui minimisent l'erreur quadratique :

$$\chi(a,b,c) = \frac{1}{N}\sum_{i=1}^{N} |y_i - (ax_i^2 + bx_i + c)|^2. \tag{3}$$

Un extremum de la fonction d'erreur χ est obtenu pour le triplet $(\hat{a},\hat{b},\hat{c})$ qui annule les trois dérivées partielles :

$$\left(\frac{\partial\chi}{\partial a}\right)_{a=\hat{a}} = 0, \qquad \left(\frac{\partial\chi}{\partial b}\right)_{b=\hat{b}} = 0, \qquad \left(\frac{\partial\chi}{\partial c}\right)_{c=\hat{c}} = 0. \tag{4}$$

On admettra que cet extremum est le minimum cherché.
On introduit les quantités :

$$S_x = \sum_{i=1}^{N} x_i \quad ; \quad S_y = \sum_{i=1}^{N} y_i \quad ; \quad S_{2x} = \sum_{i=1}^{N} x_i^2$$

$$S_{3x} = \sum_{i=1}^{N} x_i^3 \quad ; \quad S_{xy} = \sum_{i=1}^{N} x_i y_i \quad ; \quad S_{2xy} = \sum_{i=1}^{N} x_i^2 y_i \tag{5}$$

$$\text{et} \qquad S_{4x} = \sum_{i=1}^{N} x_i^4$$

Les coefficients $\hat{a}$, $\hat{b}$ et $\hat{c}$ minimisant la fonction erreur (voir eq. (3)) verifient le système linéaire suivant :

$$\begin{pmatrix} N & S_x & S_{2x} \\ S_x & S_{2x} & S_{3x} \\ S_{2x} & S_{3x} & S_{4x} \end{pmatrix} \begin{pmatrix} c \\ b \\ a \end{pmatrix} = \begin{pmatrix} S_y \\ S_{xy} \\ S_{2xy} \end{pmatrix} \tag{6}$$

On appelle Det le déterminant de la matrice 3×3 du système linéaire (6). Son expression est,

$$Det = N\big(S_{2x}S_{4x} - (S_{3x})^2\big) - S_x\big(S_xS_{4x} - S_{3x}S_{2x}\big) + S_{2x}\big(S_xS_{3x} - (S_{2x})^2\big) \tag{7}$$

Les coefficients a, b et c sont donnés par :

$$\begin{aligned}
a &= \frac{1}{Det}\Big[N\big(S_{2x}S_{2xy} - S_{xy}S_{3x}\big) - S_x\big(S_xS_{2xy} - S_{xy}S_{2x}\big) + S_y\big(S_xS_{3x} - (S_{2x})^2\big)\Big] \\
b &= \frac{1}{Det}\Big[N\big(S_{xy}S_{4x} - S_{3x}S_{2xy}\big) - S_y\big(S_xS_{4x} - S_{3x}S_{2x}\big) + S_{2x}\big(S_xS_{2xy} - S_{2x}S_{xy}\big)\Big] \\
c &= \frac{1}{Det}\Big[S_y\big(S_{2x}S_{4x} - (S_{3x})^2\big) - S_x\big(S_{xy}S_{4x} - S_{3x}S_{2xy}\big) + S_{2x}\big(S_{xy}S_{3x} - S_{2x}S_{2xy}\big)\Big]
\end{aligned} \tag{8}$$

3.2 Programmation

Dans un nouveau script `regression_3.py` :

1. Charger dans deux tableaux les abscisses et ordonnées des points contenus dans `points2.dat`.
2. Calculer les sept sommes définies par les équations (5).

3. Calculer le déterminant de la matrice du système linéaire (6).
4. Calculer les valeurs de a, b et c. Les afficher à l'écran et recopier le résultat en commentaire dans le fichier source.
5. Remplir un tableau `z` de taille `N` qui contient le résultat obtenu par la régression quadratique, c'est-à-dire
$$z_i = ax_i^2 + bx_i + c$$
6. Représenter la courbe y en fonction de x avec des points et superposer la courbe z en fonction de x avec des segments de droite. Commenter les résultats obtenus.
7. Faire de même avec la fonction `numpy.linalg.lstsq` qu'il faudra d'abord importer (elle ne fait qu'un fit linéaire).
8. Faire de même avec la fonction `numpy.polyfit` qu'il faudra d'abord importer (elle fait un fit polynômial).