1 TD 7 : Régression linéaire

(correction page ??)

Abordé lors de cette séance	
programmation	`numpy` ou calcul matriciel
algorithme	régression linéaire

Le calcul matriciel est aujourd'hui très répandu et présent dans la plupart des logiciels mathématiques gratuits tels que *R*[1], *SciLab*[2], *Octave*[3] ou payants *Gauss*[4], *Matlab*[5], *S+*[6]. Le language *Python* propose un module qui reprend le calcul matriciel proposé par tous ces langages avec des notations similaires. C'est un module qu'il faut télécharger sur Internet à l'adresse suivante : http://numpy.scipy.org/ [7]. Un tutoriel en anglais est aussi disponible à l'adresse suivante : http://www.scipy.org/Tentative_NumPy_Tutorial [8]. Ce TD appliquera le calcul matriciel à une régression linéaire [9].

Première demi-heure : données .

1) On applique le principe de la régression aux temps de parcours du marathon au fil des années [10]. Ce fichier contient quatre colonnes : villes, année, temps, temps en seconde. On peut le convertir en une matrice comme suit :

```
#coding:latin-1
import urllib, os, os.path
def charge_donnees () :
    if os.path.exists ("marathon.txt") :
        # si le fichier existe (il a déjà été téléchargé une fois)
        f = open ("marathon.txt", "r")
        text = f.read ()
        f.close ()
    else :
        # si le fichier n'existe pas
        link = "http://www.xavierdupre.fr/enseignement/td_python/" + \
                 "python_td_minute/data/regression/marathon.txt"
        url = urllib.urlopen (link)
        text = url.read ()
```

1. http://www.r-project.org/, c'est le plus utilisé par les chercheurs dans des domaines à ceux que l'ENSAE aborde.
2. http://www.scilab.org/
3. http://www.gnu.org/software/octave/
4. http://www.aptech.com/
5. http://www.mathworks.com/
6. http://spotfire.tibco.com/products/s-plus/statistical-analysis-software.aspx
7. Il faut faire attention de bien choisir la version correspondant à votre système d'exploitation (Windows, Linux, Apple) et à la version de votre langage *Python*.
8. voir aussi http://www.scipy.org/NumPy_for_Matlab_Users
9. http://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire, http://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire_multiple ou http://en.wikipedia.org/wiki/Linear_regression
10. http://www.xavierdupre.fr/enseignement/td_python/python_td_minute/data/regression/marathon.txt, ces données proviennent du blog d'Arthur Charpentier http://freakonometrics.blog.free.fr/index.php?post/2011/09/27/agacant-cette-manie-de-battre-des-records.

```
        # on enregistre les données pour éviter de les télécharger une seconde fois
        f = open ("marathon.txt", "w")
        f.write (text)
        f.close ()

    lines = text.split ("\n")
    lines = [ l.split("\t") for l in lines if len(l) > 3 ]

    # conversion en réel des données numérique
    for l in lines :
        l [1] = float(l[1])
        l [3] = float(l[3])
    return lines

if __name__ == "__main__" :
    matrice = charge_donnees ()
    print "nombre de lignes ", len(matrice)
```

2) Le premier réflexe est ensuite de dessiner les données. On utilise le module *matplotlib*[11] Pour dessiner, il est assez simple d'aller à la gallerie pour choisir le graphique[12] qui vous convient et de cliquer dessus pour voir le programme qui permet de le tracer[13].

```
# coding:latin-1
import marathon
# cette première ligne suppose que le programme de la la première question
# a été enregistrée dans un fichier marathon.py
# il n'y a alors pas besoin de le recopier ici
import matplotlib
import matplotlib.pyplot as plt

def dessin (donnees, titre = "titre") :
    x = [ d[0] for d in donnees ]
    y = [ d[1] for d in donnees ]
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(x,y, 'o')
    ax.set_title(titre)
    plt.show()

if __name__ == "__main__" :
    matrice = marathon.charge_donnees()
    mat     = [ (m[1], m[3]) for m in matrice ]
    dessin (mat)
```

Seconde demi-heure : numpy

On crée une matrice numpy comme suit :

```
from marathon import *
import numpy as np
```

11. http://matplotlib.sourceforge.net/, il faut préalablement télécharger et installer ce module.
12. http://matplotlib.sourceforge.net/gallery.html
13. http://matplotlib.sourceforge.net/examples/api/unicode_minus.html

```
donnees = charge_donnees ()
mat = np.matrix(donnees)
```

3) Qu'affichent les différentes instructions suivantes :

```
print mat.ndim
print mat.shape
print mat.size
```

On peut manipuler les matrices facilement en utilisant le symbol ":". Que permettent de faire les instructions suivantes ?

```
a = mat[0,0]
b = mat[:,0]
c = mat[0,:]
d = mat[4:10,:]
e = mat.transpose()
```

4) On peut aussi extraire deux colonnes et les coller ensemble :

```
ms = np.column_stack ( (mat[:,1], mat[:,3]))
ms = mat[:,[1,3]]  # seconde écriture
```

Quelle erreur provoque l'instruction suivante et pourquoi ?

```
ms * 3.0
```

L'instruction `ms * 3.0` ne fait rien de visible, il faudrait pour cela utiliser `print ms * 3.0` ou encore conserver le résultat dans une variable : `y = ms * 3.0`. Est-ce que la même erreur apparaît avec les deux instructions suivantes :

```
xy = np.matrix ( ms, dtype=float)
xy * 3.0
```

Complétez la ligne suivante pour calculer la moyenne des temps obtenus pour le marathon ?

```
print xy [ ... ].sum() / xy.shape [ ...]
```

5) On souhaite ajouter une colonne de zéros à la matrice `mat`, complétez le programme suivant :

```
z   = np.zeros ( mat.shape[ ... ] )
mat = np.column_stack ( (mat, z))
```

6) Que fait l'instruction suivante :

```
mat [:,4] = 1.0
```

Troisième demi-heure : régression linéaire

La régression linéaire est une façon de relier une variable Y à une autre X. On écrit le modèle suivant :

$$Y = \alpha X + \beta + \epsilon \tag{1}$$

Pour estimer les coefficients α et β, on utilise plusieurs observations (ici 360) puis on minimise l'écart ϵ_i entre la valeur observée Y_i et la valeur estimée $\hat{Y_i} = \alpha X_i + \beta$ pour chaque observation i. Si les écarts sont indépendants et distribués selon une loi normale, cela revient à minimiser l'expression suivante :

$$E = \min_{\alpha,\beta} \left(\sum_i \epsilon_i^2 \right) \tag{2}$$

$$= \min_{\alpha,\beta} \left(\sum_i \left(Y_i - (\alpha X_i + \beta) \right)^2 \right) \tag{3}$$

Afin de simplifier l'écriture du problème, on suggère de considérer une seconde variable constante $X_2 = 1$ et d'écrire :

$$\alpha X + \beta = \alpha_1 X_1 + \alpha_2 X_2 = X \left(\alpha_1, \alpha_2 \right) = XA \tag{4}$$

Où X est la matrice composée des colonnes (X_1, X_2). Dans ce cas, on résoud ce problème en annulant la dérivée et A vaut :

$$A = \left(X'X \right)^{-1} X'Y \tag{5}$$

7) Calculer A.

8) Construire une matrice contenant les trois colonnes suivantes : année, temps de course, temps de course estimé.

9) Adapter la fonction utilisée lors de la première demie-heure pour dessiner le temps de course, le temps de cours estimé en fonction des années.

Quatrième demi-heure : ajout d'une variable

10) On souhaite ajouter la dénivellation maximum de chaque marathon. Voici les altitudes minimales et maximales observées sur chaque parcours [14] :

14. source : http://www.marathonguide.com/races/races.cfm

Paris	30-70
Berlin	25-77
Amsterdam	0-10
Londres	5-40
Stockholm	0-35
Fukuoka	0-8
Boston	0-120
Chicago	172-180

Il faut remplacer le zéro de la dernière colonne par la différence entre le minimum est le maximum. On s'aidera pour cela de l'instruction suivante :

```
mat[ mat[:,0] == "PARIS" , 4 ] = nouvelle valeur
```

11) Calculer les nouveaux coefficients en considérant le modèle suivant :

$$Y = \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \epsilon \tag{6}$$

Où Y est le temps de course, X_1 est l'année, X_2 est la dénivellation, X_3 est une constante (=1).

Pour aller plus loin ou pour ceux qui ont fini plus tôt .

12) Dessiner le temps de course, le premier temps de cours estimé, le second temps de course, en fonction des années.

13) En quelle année le temps de course estimé devient-il nul ? Qu'est-ce que cela vous suggère sur la pertinence du modèle linéaire dans ce cas ?

14) Il est possible de "tester" la nullité d'un coefficient pour vérifier si une variable est importante lors de la régression [15]. En particulier pour la dénivellation, on cherche à savoir si le coefficient est presque nul ou pas. On calcule pour cela la valeur :

$$t_{n-p-1} = \frac{a_j}{\sqrt{\frac{\|y-XA\|^2}{n-p-1}(X'X)^{-1}_{jj}}} \tag{7}$$

Où n est le nombre d'observations, p la dimension du problème, M_{jj} le j^{ieme} coefficient sur la diagonale. Ce coefficient suit une loi de Student [16]. Il suffit de comparer la valeur obtenue avec la table de la loi de Student.

15. voir *Probabilités, Analyse des Données et Statistiques* de Gilles Saporta, Editions Technip.
16. http ://fr.wikipedia.org/wiki/Loi_de_Student