ENSAE TD noté, mardi 29 novembre 2011

Le programme construit au fur et à mesure des questions devra être imprimé à la fin du TD et rendu au chargé de TD. Il ne faut pas oublier de mentionner son nom en commentaires au début du programme et l'ajouter sur chaque page. Les réponses autres que des parties de programme seront insérées sous forme de commentaires. Les définitions de fonctions proposées ne sont que des suggestions.

1

Cet exercice doit être réalisé avec le module `numpy`.

Cet exercice a pour but de revenir sur le calcul matriciel proposé par le module `numpy`. Il s'appuie sur une enquête de l'INSEE réalisée en 2003 sur un échantillon du 8000 personnes [1] dont on a extrait une sous-partie que récupère le programme suivant [2]. Ce programme fournit également quelques exemples de manipulation de la classe `array` du module `numpy`.

```
#coding:latin-1
import urllib, os, os.path, numpy
def charge_donnees (nom = "donnees_enquete_2003_television.txt") :
    if os.path.exists (nom) :
        # si le fichier existe (il a déjà été téléchargé une fois)
        f = open (nom, "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/examen/" + nom
        url = urllib.urlopen (link)
        text = url.read ()
        # on enregistre les données pour éviter de les télécharger une seconde fois
        f = open (nom, "w")
        f.write (text)
        f.close ()

    lines = text.split ("\n")
    lines = [ l.split("\t") for l in lines if len(l) > 3 ]
    lines = [ [ "0" if s.strip() == "" else s for s in l ] for l in lines ]
    return lines

donnees = charge_donnees ()

colonne = donnees [0]
matrice = numpy.array (donnees [1:], dtype=float)

# quelques exemples d'utilisation du module numpy
petite_matrice = matrice [0:5,2:4]
print petite_matrice
```

1. disponible à l'adresse http://www.insee.fr/fr/themes/detail.asp?ref_id=fd-hdv03&page=fichiers_detail/HDV03/telechargement.htm

2. téléchargeable à l'adresse http://www.xavierdupre.fr/enseignement/examen_python/python_examen_2011_2012.py

```
# dimension d'une matrice
print petite_matrice.shape

# multiplication terme à terme
vecteur = petite_matrice [:,0] * petite_matrice [:,1]
print vecteur

# sum
print vecteur.sum ()

# changer une valeur selon une condition
petite_matrice [ petite_matrice [:,1] == 1, 1] = 5
print petite_matrice

# ne conserver que certaines lignes de la matrice
m = petite_matrice [ petite_matrice[:,1] == 5,: ]
print m

# créer une matrice 10x2 avec des zéros
m = numpy.zeros( (10,2) )

# trier les lignes selon la première colonne
tr = numpy.sort (petite_matrice, 0)


# dessiner deux courbes
def dessin_temperature (temperature) :
    import pylab
    u = [ t[3] for t in temperature ]
    v = [ t[4] for t in temperature ]
    pylab.plot (u)
    pylab.plot (v)
    pylab.show()
```

Le fichier téléchargé [3] est stocké dans la variable `matrice`, il contient quatre colonnes :

colonne	nom	description
0	POIDSLOG	Pondération individuelle relative
1	POIDSF	Variable de pondération individuelle
2	cLT1FREQ	Nombre d'heures en moyenne passées à regarder la télévision
3	cLT2FREQ	Unité de temps utilisée pour compter le nombre d'heures passées à regarder la télévision, cette unité est représentée par les quatre valeurs suivantes : 0 non concerné 1 jour 2 semaine 3 mois

Le fichier contient des lignes comme celles qui suivent. Sur la première ligne contenant des chiffres, un individu a déclaré passer deux heures par jour devant la télévision. Sur la suivante, un autre individu a déclaré passer six heures par semaine.

3. http://www.xavierdupre.fr/enseignement/td_python/python_td_minute/data/examen/donnees_enquete_2003_television.txt

```
POIDLOG       POIDSF         cLT1FREQ  cLT2FREQ
0.8894218317  4766.8652013   2         1
2.740069772   14685.431344   6         2
```

Après la première exécution, le fichier est présent sur votre disque dur à côté du programme, il est alors facile de l'ouvrir de voir ce qu'il contient.

1) Toutes les lignes ne sont pas renseignées. Modifier la matrice pour enlever les lignes pour lesquelles l'unité de temps (cLT2FREQ) n'est pas renseignée ou égale à zéro. (1 point)

2) Remplacer les valeurs de la quatrième colonne par une durée en heures. (1 point)

3) Calculer le nombre d'heures par jour moyen passées devant la télévision et écrire la réponse en commentaire de la fonction. (1 point)

4) Calculer ce même nombre mais pondéré par la colonne POIDSF et écrire la réponse en commentaire de la fonction. (2 points)

5) Combien de gens dans l'échantillon regardent-ils la télévision plus de 24h par jour ? Quelle est la raison la plus probable ? Ecrire la réponse en commentaire de la fonction. (1 point)

6) Calculer la médiane non pondérée. Ecrire la réponse en commentaire de la fonction. (2 points)

2

Cet exercice peut-être réalisé indifféremment avec le module `numpy` ou non.

Mes grands-parents passent chaque année l'été à Charleville-Mézières (dans le Nord-Est de la France) et l'hiver à Cannes (dans le Sud). Les températures mesurées quotidiennement sur l'année 2010 sont représentées par la figure 1 [4]. Mes grands-parents cherchent à se rapprocher d'une température idéale de 20 degrés. Ils voudraient savoir quand ils auraient dû partir à Cannes et quand en revenir pour atteindre le plus possible cet objectif en 2010.

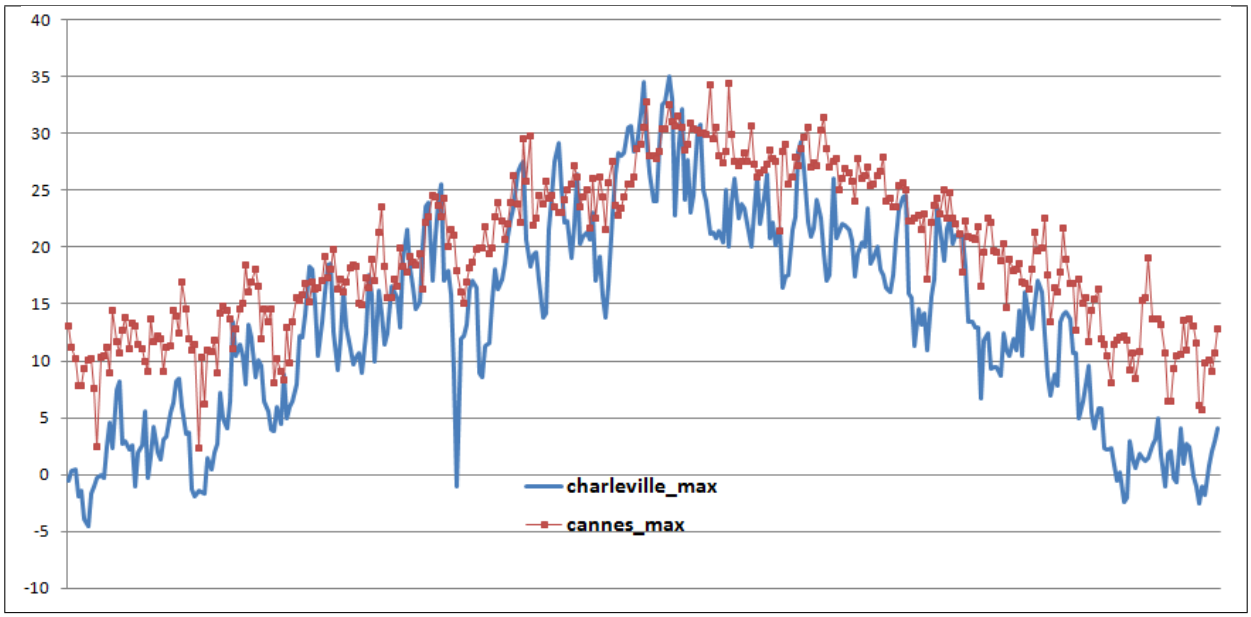

Figure 1 : *Températures maximales mesurées quotidiennement à Cannes et Charleville-Mézières en 2010.*

L'exercice précédent utilise une fonction `charge_donnees` qu'on utilise également pour récupérer les données de températures [5].

```
temperature = charge_donnees("cannes_charleville_2010_max_t.txt")
```

Après la première exécution, le fichier est présent sur votre disque dur à côté du programme, il est alors facile de l'ouvrir de voir ce qu'il contient. L'ensemble peut être aussi copié/collé sous Excel.

1) Ecrire une fonction qui récupère le tableau matrice et convertit tous les éléments en réels (faire attention à la première ligne). (1 point)

2) Certaines valeurs sont manquantes et sont remplacées par la valeur -1000 choisie par convention (voir la date du 4 mai). Ecrire une fonction qui remplace ces valeurs par la moyenne des deux valeurs adjacentes. (3 points)

On pourra s'aider de la fonction suivante pour vérifier que les manquantes ne sont plus présentes :

4. Ces données sont extraites du site http://www.meteociel.fr/climatologie/villes.php.
5. http://www.xavierdupre.fr/enseignement/td_python/python_td_minute/data/examen/cannes_charleville_2010_max_t.txt

```
def dessin_temperature (temperature) :
    import pylab
    u = [ t[3] for t in temperature ]
    v = [ t[4] for t in temperature ]
    pylab.plot (u)
    pylab.plot (v)
    pylab.show()
```

3) Après avoir discuté avec mes grands-parents, nous avons décidé d'utiliser une fonction de coût égale au carré de l'écart entre deux températures : la température qu'ils souhaitent (20 degré) et la température qu'ils ont (celle de Charleville ou Cannes). Ecrire une fonction `distance` qui calcule la fonction suivante : (1 point)

$$d(T_1, T_2) = (T_1 - T_2)^2 \tag{1}$$

4) Supposons que nous connaissons les deux dates d'aller t_1 et de retour t_2, mes grands-parents seraient donc à :

hiver	t_1	été	t_2	hiver
Cannes		Charleville		Cannes

Ecrire une fonction qui calcule la somme des écarts de température au carré entre la température T souhaitée et la température où ils sont : (3 points)

$$E = \sum_{t=1}^{t=t_1} d\left(T(Cannes, t), T\right) + \sum_{t=t_1}^{t=t_2} d\left(T(Charleville, t), T\right) + \sum_{t=t_2}^{t=365} d\left(T(Cannes, t), T\right) \tag{2}$$

5) Ecrire une fonction qui détermine les meilleures valeurs t_1 et t_2. Ecrire en commentaire de votre programme le raisonnement suivi et la réponse trouvée. (3 points)

6) Quel est le coût de votre algorithme en fonction du nombre de jours ? Si, on remplace la série de températures quotidienne par une série hebdomadaire, l'algorithme est combien de fois plus rapide ? (1 point)

Correction

```
# coding: latin-1
# ce fichier contient le programme fournit au début de l'examen
# http://www.xavierdupre.fr/enseignement/examen_python/python_examen_2011_2012.py
from td_note_2012_enonce import *
import numpy, pylab

############################################################################
# exercice 1
############################################################################

# question 1 (+1=1p)
donnees = charge_donnees ()
colonne = donnees [0]
matrice = numpy.array (donnees [1:], dtype=float)

mat     = matrice
mat     = mat [ mat [:,3] > 0, : ]
```

```
# question 2 (+1=2p)
mat [ mat[:,3] == 1, 3 ] = 24
mat [ mat[:,3] == 2, 3 ] = 24*7
mat [ mat[:,3] == 3, 3 ] = 24*30

# question 3 (+1=3p)
res = mat [:,2] / mat [:,3]
print res.sum() / res.shape [0]       # 0.111 ~ 11,1% du temps passé devant la télévision
print res.sum() / res.shape [0] * 24  # soit 2h40min

# question 4 (+2=5p)
m = mat[:,1] * mat[:,2] / mat[:,3]
print m.sum() / mat[:,1].sum()        # 0.108 ~ 10,8%

# question 5 (+1=6p)
m = mat[ mat[:,2] > mat[:,3], : ]
print m  # il y a deux personnes et la raison la plus probable est une erreur dans l'unité de temps

# question 6 (+2=8p)
res = numpy.sort (res, 0)
print res[res.shape[0]/2]   # 0.083 ~ 8.3% = 2h

# question 7 (+2=10p)
pr = numpy.zeros ((mat.shape[0],4))
pr [:,0] = mat[:,2] / mat[:,3]
pr [:,1] = mat[:,1]
pr [:,2] = pr[:,0] * pr[:,1]
pr = numpy.sort (pr, 0)
total = pr[:,2].sum()
pr[0,3] = pr [0,2]
for i in xrange (1, pr.shape[0]) :
    pr[i,3] = pr[i-1,3] + pr[i,2]
    if pr[i,3]/total > 0.5 :
        fin = i
        break
print pr[fin,3] / pr[:fin+1,1].sum()  # 0.0895 ~ 8.95%

##########################################################################
# exercice 2
##########################################################################

# question 1 (+1=1p)
temperature = charge_donnees("cannes_charleville_2010_max_t.txt")

def conversion_reel (temperature) :
    return [ [ float (x) for x in l ] for l in temperature [1:] ]

temperature = conversion_reel(temperature)

#question 2  (+2=3p)
def valeur_manquante (temperature, c) :
    for i in xrange (1, len (temperature)-1) :
        if temperature [i][c] == -1000 :
            temperature [i][c] = (temperature [i-1][c] + temperature [i+1][c]) / 2

valeur_manquante(temperature, 3)
valeur_manquante(temperature, 4)
```

```
def dessin_temperature (temperature) :
    import pylab
    u = [ t[3] for t in temperature ]
    v = [ t[4] for t in temperature ]
    pylab.plot (u)
    pylab.plot (v)
    pylab.show()

# on met en commentaire pour éviter de l'exécuter à chaque fois
# dessin_temperature(temperature)

# question 3 (+1=4p)

def distance (u,t) :
    return (u-t)**2

# question 4 (+3=7p)

def somme_ecart (temperature, t1, t2, T) :
    s = 0
    for i in xrange (0, len(temperature)) :
        if t1 < i < t2 :
            s += distance (temperature[i][3], T) # charleville
        else :
            s += distance (temperature[i][4], T) # cannes
    return s

# question 5 (+3=10p)

def minimisation (temperature, T) :
    best = 1e10
    t1t2 = None
    for t1 in xrange (0,len(temperature)) :
        for t2 in xrange (t1+1,len(temperature)) :
            d = somme_ecart(temperature,t1,t2,T)
            if best == None or d < best :
                best = d
                t1t2 = t1,t2
    return t1t2, best

#for i in range (300,363) : print "*",somme_ecart (temperature, i, i+2, 20)
print temperature [191]
print temperature [266]
for T in range (15, 25) :
    print T, "**",minimisation (temperature,T) # (191 = 11/7, 266 = 24/9)  (attendre 2 minutes)

# question 6
# Le coût de l'algorithme est on O(n^2) car l'optimisation est une double boucle sur les températures.
# Passer des jours aux semaines, c'est utiliser des séries 7 fois plus courtes,
# l'optimisation sera 7^2 fois plus rapide.
```