1 Correction du TD ??, page ??

1)

```
def random_set (nb = 100) :
    res = []
    for i in range (0, nb) :
        x,y = random.gauss (0,1),random.gauss (0,1)
        res.append ([x,y])
    for i in range (0, nb*2) :
        x,y = random.gauss (0,1),random.gauss (0,1)
        n   = (x**2 + y**2) ** 0.5
        if n == 0 : n == 1.0
        x  *= 5.0 / n
        y  *= 5.0 / n
        x  += random.gauss (0,0.5)
        y  += random.gauss (0,0.5)
        res.append ([x,y])
    return res
```

2)

3) 4) 5) 6)

7) Résultat de la classification, figure 1.

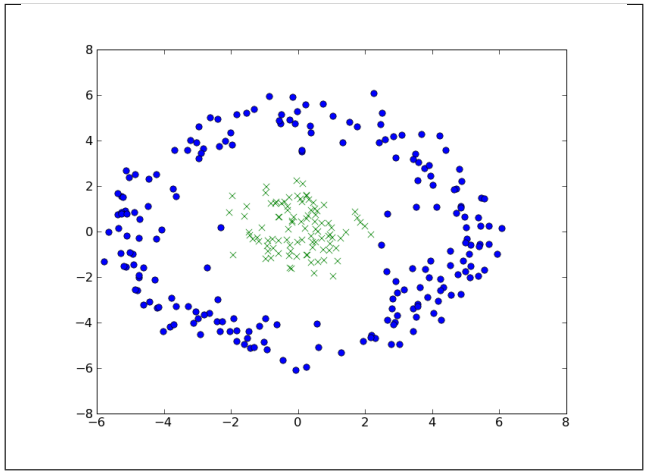

Figure 1 : *Résultat de la classification du nuage de points figure* **??**.

```
# coding: latin-1
import sys,random,copy,math
import matplotlib.pyplot as plt
import numpy as np

def random_set (nb = 100) :
    """construit un échantillon aléatoire avec deux cercles concentriques,
    nb pour le premier, nb*2 pour le second"""
    res = []
```

```
    for i in range (0, nb) :
        x,y = random.gauss (0,1),random.gauss (0,1)
        res.append ([x,y])
    for i in range (0, nb*2) :
        x,y = random.gauss (0,1),random.gauss (0,1)
        n   = (x**2 + y**2) ** 0.5
        if n == 0 : n == 1.0
        x  *= 5.0 / n
        y  *= 5.0 / n
        x  += random.gauss (0,0.5)
        y  += random.gauss (0,0.5)
        res.append ([x,y])
    res.sort ()
    return res

def draw (points, clas = None) :
    """dessine un nuage de points, si clas est une liste,
    elle contient un indice de clas"""

    if clas == None :
        fig = plt.figure()
        ax = fig.add_subplot(111)
        x = [ p [0] for p in points ]
        y = [ p [1] for p in points ]
        ax.plot (x,y, 'o')
        plt.savefig ("im1.png")
    else :
        fig = plt.figure()
        ax = fig.add_subplot(111)
        x = [ p [0] for p,c in zip (points, clas) if c == 0 ]
        y = [ p [1] for p,c in zip (points, clas) if c == 0 ]
        ax.plot (x,y, 'o')
        x = [ p [0] for p,c in zip (points, clas) if c == 1 ]
        y = [ p [1] for p,c in zip (points, clas) if c == 1 ]
        ax.plot (x,y, 'x')
        plt.savefig ("im2.png")

def distance_ligne (mat) :
    """retourne une matrice dont chaque case correspond aux distances entre lignes"""
    prod = mat * mat.T
    dist = copy.deepcopy (prod)
    lin  = dist.shape [0]

    di = np.diag (prod)
    di = np.matrix (di)
    one  = np.ones ((1,lin))
    ii   = one.transpose () * di
    jj   = di.transpose () * one
    dist = prod * (-2) + ii + jj

    def sqrt (x) : return x**0.5 if x >= 0 else 0.0
    func_sqrt = np.vectorize (sqrt, otypes=[float])
    dist = func_sqrt (dist)

    # autre essai
    #def m (x)    : return x**0.6 if x >= 0 else 0.0
    #func_m    = np.vectorize (m, otypes=[float])
    #dist = func_m (dist)
```

```
    #code dont la logique est plus explicite mais il est beaucoup plus lent
    #for i in xrange (0, lin) :
    #    for j in xrange (0, lin) :
    #        x = (prod [i,i] + prod [j,j] - 2*prod [i,j])
    #
    #        if x <= 0 : dist [i,j]= 0  #problème d'arrondi numérique
    #        else : dist [i,j]= x**0.5

    return dist

def iteration (dist) :
    """itération de l'algorithme"""
    dist = distance_ligne (dist)
    lin  = dist.shape [0]
    for i in xrange (0, lin) :
        x = np.max (dist [i,:])
        y = dist [i,:] * (1.0 / x )#* lin)
        dist [i,:] = y
    return dist

def algorithme_cluster (points) :
    """algorithme"""
    mat     = np.matrix (points)
    lin,col = mat.shape
    dist    = distance_ligne (mat)
    for i in range (0,50) :
        print "itération i", i, np.min (dist [0,1:]), np.max (dist), np.sum (dist [0,:])
        dist = iteration (dist)

    M    = np.max (dist [0,:])/2
    res  = dist [0,:]
    good = res > M
    bad  = res <= M
    res [good]= 1
    res [bad] = 0
    li   = res.tolist () [0]

    return li

if __name__ == "__main__" :

    # construction of the random set (two circles, a line)
    rnd = random_set ()
    #draw (rnd)
    clas = algorithme_cluster (rnd)
    draw (rnd, clas)
    plt.show ()
```

8) A propos de la partie financière qui ne traite pas le problème des décalages entre les dates lors du téléchargement des données mais qui le détecte.

```
# coding: latin-1
import sys
import random
import matplotlib.pyplot as plt
import numpy as np
import copy
```

```
import math
import os
from cluster import *

import urllib
import datetime
import sys

def get_cac40_quotes () :
    """récupère les cotes du CAC40 depuis un fichier texte
    format : quote \t nom
    """
    file  = "cac40_quote.txt"
    quote = open (file, "r").readlines ()
    quote = [ q.strip (" \n\r") for q in quote ]
    quote = [ q.split ("\t") for q in quote if len (q) > 0 ]
    assert len (quote) == 40
    return quote

def get_date (s) :
    """convertit une date depuis une chaîne de caractères vers un format Python"""
    y,m,d = s.split ("-")
    y,m,d = int(y),int(m),int(d)
    d     = datetime.datetime (y,m,d)
    return d

def download_quotes (code, d1, d2) :
    """
    télécharge la cotation d'une action entre deux dates
        - code: cote
        - d1:   première date (format python)
        - d2:   seconde date (format python)
    """
    root = "http://ichart.yahoo.com/table.csv?s=%s&a=%02d&b=%d&c=%d" \
                                              "&d=%02d&e=%d&f=%d&g=d&ignore=.csv" % \
               (code, d1.month-1, d1.day, d1.year, d2.month-1, d2.day, d2.year)

    f    = urllib.urlopen (root)
    qu   = f.read ()
    f.close ()
    lines = qu.split ("\n")
    lines = [ l .strip (" \n\r") for l in lines ]
    head  = lines [0]
    lines = lines [1:]
    lines = [ l.split (",") for l in lines if "," in l ]
    lines = [ (get_date (l[0]), l) for l in lines ]
    lines = [ l [1] for l in lines if d1 <= l [0] <= d2 ]
    lines = [ "\t".join (l) for l in lines ]

    return "\n".join ([head ] + lines)

def get_quotes_internet (all) :
    """télécharge toutes les cotations pour les cotes dans all pour les 6 derniers mois
        - all: dictionnaire { cote: nom complet }
        - enregistre le résultat dans un fichier pour éviter de télécharger
            les cours à chaque exécution
    """
    year2  = datetime.datetime (2009,1,1) - datetime.datetime (2008,6,1)
    today = datetime.datetime (2009,1,1).now ()
```

```
    lyear = today - year2
    path  = "quotes/"
    for a,nom in all :
        file = path + a + ".txt"
        if os.path.exists (file) : continue
        res = download_quotes (a, lyear, today)
        f = open (file, "w")
        f.write (res)
        f.close ()

        print "loading ", a, " from ", lyear , " to ", today, " lines ", len (res.split ("\n"))

def get_close_ret (code) :
    """retourne la série des rendements depuis un fichier de cotations
    créé par la fonction get_quotes_internet"""
    file = "quotes/" + code + ".txt"
    lines = open (file, "r").readlines ()
    lines = lines [1:]
    lines = [ l for l in lines if "\t" in l ]
    lines = [l.strip ("\n\r ").split ("\t") [4] for l in lines ]
    lines = [ float (x) for x in lines ]
    lines = [ (lines [x] - lines [x-1]) / lines [x-1] for x in xrange (1, len (lines)) ]
    return lines

def get_matrix (all) :
    """retourne la matrice des autocorrélations"""
    colonnes = []
    for a,nom in all :
        close = get_close_ret (a)
        if len (colonnes) != 0 and len (close) != len (colonnes [0]) :
            message = "problème avec %s longueur %d <> %d " % (nom, len (close), len (colonnes [0]))
            raise Exception ("les données ne sont pas alignées dans le temps, " \
                             "une série ne contient le même nombre d'observations\n" + message)
        colonnes.append (close)
    mat = np.matrix (colonnes)  # s'il y a une erreur ici, cela signifie que les actions n'ont pas
                                # des cotations aux mêmes dates
    cor = np.corrcoef (mat)
    return cor


if __name__ == "__main__" :

    if not os.path.exists ("quotes"): os.mkdir ("quotes")
    quotes = get_cac40_quotes ()
    get_quotes_internet (quotes)
    mat = get_matrix (quotes)
    print mat.shape
    li  = algorithme_cluster (mat)
    print li
    for a,b in zip (quotes,li) :
        print b, a [1], "\t", a [0]
```

fin correction TD ?? □