TP1 – M1 IAA 2013-2014

1. Prise en main de python. Manipulation de listes.

Nous considérons que vous avez lu et fait les exercices du mini tutoriel sur python, disponible sur Moodle.

Démarrez l'interpréteur ipython en ligne de commandes en tapant dans une console (ouvrez une « Konsole » sous linux) :

```
ipython --pylab
```

Rappel du mini-tutoriel : IPython est un interpréteur plus interactif que python simple. L'option --pylab permet de charger les librairies numpy et matplotlib et active le mode graphes interactifs. Sous python normal, le mode graphes interactifs est activé en appelant une seule fois dans votre programme ou session la fonction `ion()`. De même dans python, pour importer numpy et matplotlib, il faut inclure en début de programme ou de session la commande :

```
from pylab import *
```

Dans l'interpréteur ipython, entrez la liste de listes suivante :

```
obs = [[1.0,1.0],[3.0,3.0],[1.0,5.0],[-3.0,5.0],[-5.0,3.0],[-3.0,1.0],
[3.0,-1.0],[5.0,-3.0],[3.0,-5.0],[-1.0,-5.0],[-3.0,-3.0],[-1.0,-1.0]]
```

puis essayez par exemple :

```
type(obs)
```

Quand vous tapez `obs.` et que vous appuyez sur la touche tabulation, vous obtenez les méthodes disponibles pour le type de la variable obs. Le liste obs correspond à l'exemple donné au TD1.

Pour se familiariser avec les listes, voici une série d'instructions très simples à tester :

```
l1=[]
l1.append(1)
l1
l1.append(3)
l1.append(2)
l1.append(3)
l1.count(3)
len(l1)
l1[0]
l1[1]
l1.remove(3)
l1.append([])
l1
len(l1)
l1.insert?

l2 = [[[1.0,1.0], [3.0,3.0]] , [[1.0,5.0]]]
len(l2)
l2
l2[0]
l2[1]
l2.append([9.0, 9.0])
l2
```

Il y a plusieurs manières de copier une liste dans une autre avec des conséquences différentes. Tapez ce qui suit :

```
l1=[1, 2, 3]
l2=l1
l3=list(l1)
l1.insert(0,9)
l1
l2
l3
```

Retenez que pour obtenir une copie indépendante de la liste originale, l'une des manières est d'utiliser l'instruction `list()`

```
liste_copie = list(liste_orig)
```

Revenons à la liste du TD. Tapez :

```
point1=obs[0]
point2=obs[1]
```

Puis calculez la distance euclidienne entre ces deux points.

2. Ouvrez dans un éditeur de texte (par exemple gedit, kate, ou emacs), l'embryon du programme kmeans.py
Vous pouvez exécuter ce programme dans ipython en tapant

```
run kmeans
```

Le programme ne fonctionne pas correctement après la première itération à cause du code qu'il vous faut ajouter dans la fonction `kmeans` de ce programme. Pour pouvoir prendre en compte les modifications que vous apportez au programme, avant de l'exécuter, il vous faut le ré-importer en tapant :

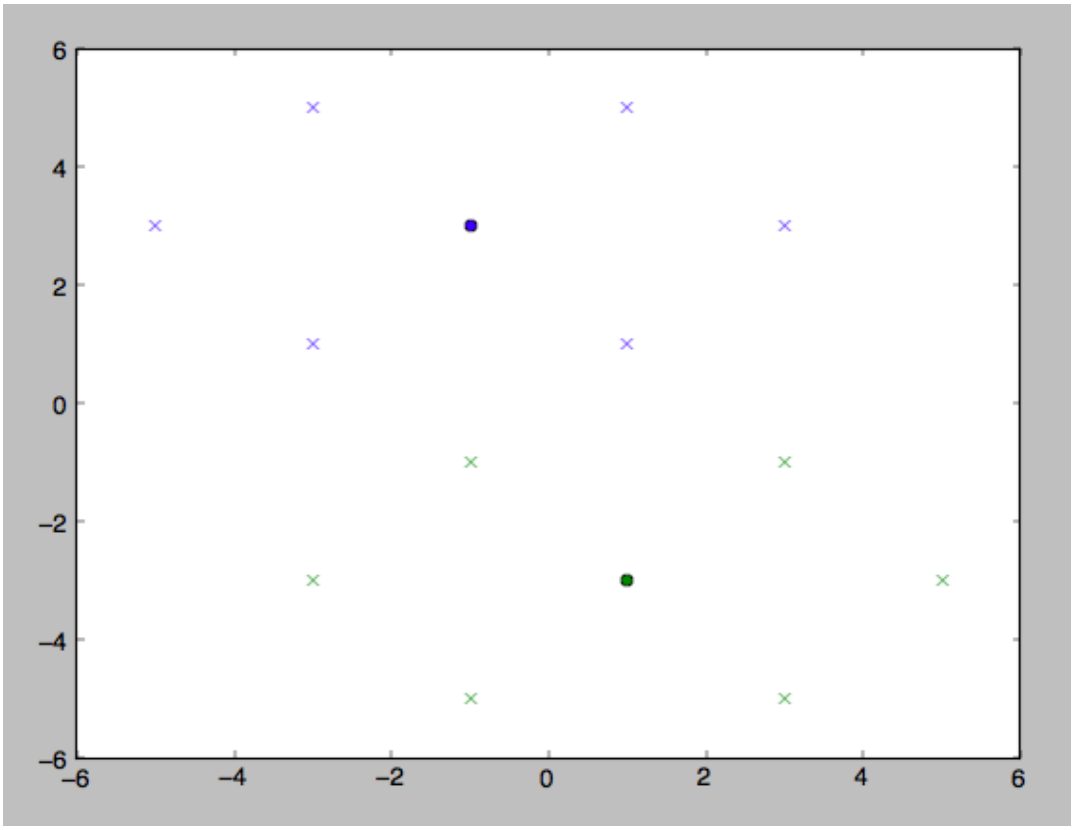
```
import kmeans
```

Complétez la fonction `kmeans` en implémentant :

1) la réallocation : il faut redéfinir la liste de listes `data` en attribuant pour chaque point de `data` un nuage de points (un nuage = une sous-liste de `data`).
2) Actualiser les centroïdes des nouveaux nuages de points

3. Vérifiez que l'initialisation aléatoire des centroïdes a un effet sur les nuages de points que l'on obtient. Améliorez le programme pour qu'il réalise une initialisation plus robuste : implémentez la fonction `init_centres(liste, nbG)`

Vous devez obtenir la figure suivante.



4. Implémentez le corps de la fonction `calcul_inertie(liste, centres)` et modifiez le critère d'arrêt de l'algorithme pour utiliser l'inertie (déjà implémenté mais commenté)

Rappel : inertie intra-classe=Somme_nuages (Somme_points_du_nuage (d(point, centroïde)^2))

5. Implémentez une méthode qui permette de déterminer le nombre de classes optimal.

Si vous désirez fonctionner avec plus de données vous pouvez importer celles qui sont dans data.txt de cette façon :

```
data = list(loadtxt("data.txt"))
     for i in range(len(data)):
          data[i] = list(data[i])
```