INF 413 Compte Rendu TP5-8
--Algorithme des K-Means

SHEN Sheng

11/10/ 2016

Sommaire

1. INTRODUCTION

Le partitionnement en k-means est une méthode de partitionnement de données et un problème d'optimisation combinatoire. Étant donnés des points et un entier k, le problème est de diviser les points en k groupes, souvent appelés clusters, de façon à minimiser une certaine fonction. On considère la distance d'un point à la moyenne des points de son cluster ; la fonction à minimiser est la somme des carrés de ces distances.

2. ETUDE DE L'ALGORITHME K-MEANS

2.1 DEFINITION

Étant donné un ensemble de points ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), on cherche à partitionner les n points en k ensembles $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ ($k \leq n$) en minimisant la distance entre les points à l'intérieur de chaque partition :

$$arg \ \min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2$$

où $\boldsymbol{\mu}_i$ est la moyenne des points dans S_i.

2.2 PRINCIPALE

On suppose qu'il existe K classes distinctes. On commence par désigner K centres de classes μ_1, ..., μ_k parmi les individus. Ces centres peuvent être soit choisis par l'utilisateur pour leur "représentativité", soit désignés aléatoirement. On réalise ensuite itérativement les deux étapes suivantes :

– Pour chaque individu qui n'est pas un centre de classe, on regarde quel est le centre de classe le plus proche. On définit ainsi K classes C_1, ..., C_k, où

C_i= {ensemble des points les plus proches du centre μ_i}.

– Dans chaque nouvelle classe C_i, on définit le nouveau centre de classe μ_i comme étant le barycentre des points de C_i. L'algorithme s'arrête suivant un critère d'arrêt fixé par l'utilisateur qui peut être choisi parmi les suivants : soit le nombre limite d'itérations est atteint, soit l'algorithme a convergé, c'est-à-dire qu'entre deux itérations les classes formées restent les mêmes, soit l'algorithme a "presque" convergé, c'est-à-dire que l'inertie intra-classe ne s'améliore quasiment plus entre deux itérations.

2.3 EXEMPLE

La figure 2.1 illustre l'algorithme sur un exemple où quatre points a (-1,1), b (0,1), c (3,0) et d (3,-1) doivent être classés en 2 classes. On remarque sur cet exemple que bien qu'à l'initialisation les centres de classes sont mal répartis, l'algorithme a convergé en retrouvant les "vraies" classes

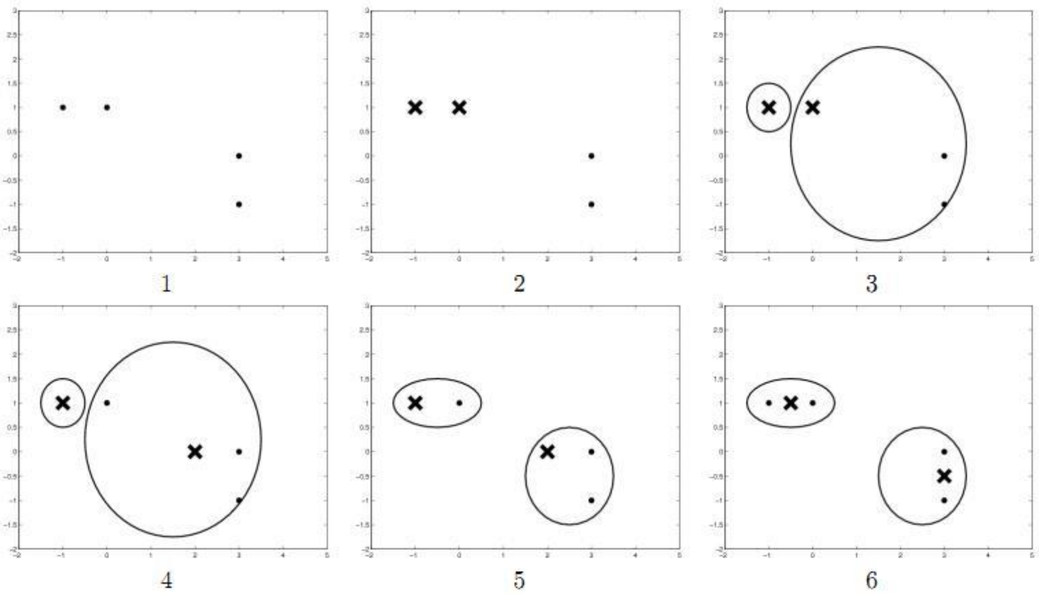

Figure 2.1 – Une illustration de l'algorithme k-means. (1) On dispose de 4 points à classer en deux classes. (2) A l'initialisation, deux de ces points sont choisis comme centres de classe. (3) Deux classes sont créées en regroupant les autres points en fonction du centre de classe le plus proche. (4) On définit les nouveaux centres de classe comme étant le barycentre des classes nouvellement créées. (5) On regroupe à nouveau les points. (6) On définit les nouveaux centres de classes. A l'étape suivante rien ne change, l'algorithme a convergé.

3. IMPLEMENTATION DE L'ALGORITHME K-MEANS

Au niveau de langage Python, on a réalisé ce TP avec choix différents ci-dessous.

On a déjà utilisé la fonction ***sys.argv[]***, qui est un liste dans le Python et qui contient les arguments de ligne de commande passés au script. Avec la fonction ***len (sys.argv)*** on peut compter le nombre d'arguments.

3.1 CHOIX DE "K"

Pour choisir le 'K', l'utilisateur peut input un numéro pour définir K en utilisant ***sys.argv[1]***. S'il égale 'random', il va générer un numéro aléatoire entre la valeur minimum 2 et la valeur maximum K. Sinon, l'utilisateur peut déterminer le K par saisir .

3.2 CHOIX DE LA DISTANCE

3.2.1 La distance Euclidienne

Pour calculer la distance, on utilise la distance Euclidienne. Soit deux points $(x_1, x_2 \dots x_n)$ et $(y_1, y_2 \dots y_n)$, la distance Euclidienne (x,y) est la somme des carrés de ces distances.

$$\boldsymbol{distance(x,y)} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3.2.2 La distance Manhattan

Pour calculer la distance Manhattan. Soit deux points $(x_1, x_2 \dots x_n)$ et $(y_1, y_2 \dots y_n)$, la distance Manhattan (x,y) est la somme des carrés de ces distances.

$$\boldsymbol{distance(x,y)} = \sum_{i=1}^{n} |x_i - y_i|$$

3.2.3 La distance Minkowski

Pour calculer la distance Minkowski. Soit deux points $(x_1, x_2 \dots x_n)$ et $(y_1, y_2 \dots y_n)$, la distance Minkowski (x,y) est la somme des carrés de ces distances.

$$\boldsymbol{distance(x,y)} = \sqrt[p]{\sum_{i=1}^{n} (x_i - y_i)^p}$$

3.3 CHOIX DU CRITERE D'ARRET

Pour le choix du critère d'arrêt, on utilise une condition pour le déterminer (on limite la durée d'exécution). Quand le nombre de changement de points est supérieur à certain chiffre, le programme va terminer. En fait, quand le nombre de point arrive à 10000, l'algorithme exécute trop long, on pense qu'il termine.

3.4 GENERER LES CENTRES

Pour gérer le choix de centres, soit on obtient les premiers K datas à partir de tous les datas, soit on générer k datas au hasard à partir de tous les datas.

3.5 METTRE À JOUR LES CENTRES

Pour mettre à jour les centres, on définit la fonction calculer les moyennes pour chaque groupe.

4. TESTS SUR DIFFERENTS JEUX DE DONNES

4.1 TEST SUR IRIS

On a supprimé les chaines de caractères de la dernière colonne d'IRIS pour faciliter programmation.

On a renvoyé le résultat dans un fichier, les 2 schémas ci-dessous sont 2 résultats que l'on a obtenus. (Le premier est IRIS original, et le deuxième est nouveau data correspond). La dernière colonne du résultat est le numéro de groupe de data et on utilise la distance Euclidienne.

On a comparé l'influence de type de la distance sur le partitionnement d'Iris data. On utilise seulement le premier k lignes en les centres. On a trouvé qu'il existe une petite erreur que n'est pas forcément à éviter, peut-être une erreur systématique.

Selon les figures, on peut constater que les type de la distance n'ont pas d'influence sur le partitionnement d'Iris data car le numéro de groupe de data est toujours les mêmes.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
5.5,3.5,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
4.4,3.0,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5.0,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
```

Figure 4.1-1 IRIS DATA ORIGINAL

```
6.235227272727272,3.2977272727272746,4.07159090909091,1.3477272727272722,0
5.641176470588237,2.729411764705882,3.7921568627450983,1.1647058823529408,1
4.981818181818182,3.4909090909090916,1.4727272727272727,0.21818181818181823,2
4.6,3.1,1.5,0.2,2
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,2
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,2
4.9,3.1,1.5,0.1,1
5.4,3.7,1.5,0.2,0
4.8,3.4,1.6,0.2,2
4.8,3.0,1.4,0.1,1
4.3,3.0,1.1,0.1,2
5.8,4.0,1.2,0.2,0
5.7,4.4,1.5,0.4,0
5.4,3.9,1.3,0.4,0
5.1,3.5,1.4,0.3,0
5.7,3.8,1.7,0.3,0
5.1,3.8,1.5,0.3,0
5.4,3.4,1.7,0.2,0
5.1,3.7,1.5,0.4,0
4.6,3.6,1.0,0.2,2
5.1,3.3,1.7,0.5,0
4.8,3.4,1.9,0.2,0
5.0,3.0,1.6,0.2,1
5.0,3.4,1.6,0.4,0
5.2,3.5,1.5,0.2,0
5.2,3.4,1.4,0.2,0
4.7,3.2,1.6,0.2,2
4.8,3.1,1.6,0.2,1
5.4,3.4,1.5,0.4,0
5.2,4.1,1.5,0.1,0
5.5,4.2,1.4,0.2,0
4.9,3.1,1.5,0.1,1
5.0,3.2,1.2,0.2,1
5.5,3.5,1.3,0.2,0
4.9,3.1,1.5,0.1,1
4.4,3.0,1.3,0.2,2
5.1,3.4,1.5,0.2,0
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,1
4.4,3.2,1.3,0.2,2
5.0,3.5,1.6,0.6,0
5.1,3.8,1.9,0.4,0
4.8,3.0,1.4,0.3,1
5.1,3.8,1.6,0.2,0
4.6,3.2,1.4,0.2,2
5.3,3.7,1.5,0.2,0
5.0,3.3,1.4,0.2,0
7.0,3.2,4.7,1.4,0
6.4,3.2,4.5,1.5,0
```

Figure 4.1-2 IRIS DATA NOUVEAU_EUCLIDIENNE

```
5.921739130434782,3.610869565217391,3.0913043478260867,0.9847826086956519,0
6.089010989010988,2.827472527472528,4.467032967032968,1.4527472527472527,1
4.976923076923077,3.4153846153846157,1.4769230769230768,0.22307692307692312,2
4.6,3.1,1.5,0.2,2
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,2
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,1
4.9,3.1,1.5,0.1,1
5.4,3.7,1.5,0.2,0
4.8,3.4,1.6,0.2,2
4.8,3.0,1.4,0.1,1
4.3,3.0,1.1,0.1,2
5.8,4.0,1.2,0.2,0
5.7,4.4,1.5,0.4,0
5.4,3.9,1.3,0.4,0
5.1,3.5,1.4,0.3,0
5.7,3.8,1.7,0.3,0
5.1,3.8,1.5,0.3,0
5.4,3.4,1.7,0.2,0
5.1,3.7,1.5,0.4,0
4.6,3.6,1.0,0.2,2
5.1,3.3,1.7,0.5,0
4.8,3.4,1.9,0.2,2
5.0,3.0,1.6,0.2,1
5.0,3.4,1.6,0.4,0
5.2,3.5,1.5,0.2,0
5.2,3.4,1.4,0.2,0
4.7,3.2,1.6,0.2,2
4.8,3.1,1.6,0.2,1
5.4,3.4,1.5,0.4,0
5.2,4.1,1.5,0.1,0
5.5,4.2,1.4,0.2,0
4.9,3.1,1.5,0.1,1
5.0,3.2,1.2,0.2,2
5.5,3.5,1.3,0.2,0
4.9,3.1,1.5,0.1,1
4.4,3.0,1.3,0.2,2
5.1,3.4,1.5,0.2,0
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,2
4.4,3.2,1.3,0.2,2
5.0,3.5,1.6,0.6,0
5.1,3.8,1.9,0.4,0
4.8,3.0,1.4,0.3,1
5.1,3.8,1.6,0.2,0
4.6,3.2,1.4,0.2,2
5.3,3.7,1.5,0.2,0
5.0,3.3,1.4,0.2,0
7.0,3.2,4.7,1.4,0
6.4,3.2,4.5,1.5,0
```

Figure 4.1-3 IRIS DATA NOUVEAU_MANHATTAN

```
6.060000000000002,3.076153846153847,4.134615384615387,1.3553846153846159,0
5.281818181818181,3.2818181818181817,1.5818181818181818,0.2,1
5.033333333333333,3.533333333333333,1.4444444444444446,0.2222222222222222,2
4.6,3.1,1.5,0.2,2
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,2
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,2
4.9,3.1,1.5,0.1,1
5.4,3.7,1.5,0.2,0
4.8,3.4,1.6,0.2,0
4.8,3.0,1.4,0.1,1
4.3,3.0,1.1,0.1,2
5.8,4.0,1.2,0.2,0
5.7,4.4,1.5,0.4,0
5.4,3.9,1.3,0.4,0
5.1,3.5,1.4,0.3,0
5.7,3.8,1.7,0.3,0
5.1,3.8,1.5,0.3,0
5.4,3.4,1.7,0.2,0
5.1,3.7,1.5,0.4,0
4.6,3.6,1.0,0.2,2
5.1,3.3,1.7,0.5,0
4.8,3.4,1.9,0.2,0
5.0,3.0,1.6,0.2,1
5.0,3.4,1.6,0.4,0
5.2,3.5,1.5,0.2,0
5.2,3.4,1.4,0.2,0
4.7,3.2,1.6,0.2,1
4.8,3.1,1.6,0.2,1
5.4,3.4,1.5,0.4,0
5.2,4.1,1.5,0.1,0
5.5,4.2,1.4,0.2,0
4.9,3.1,1.5,0.1,1
5.0,3.2,1.2,0.2,1
5.5,3.5,1.3,0.2,0
4.9,3.1,1.5,0.1,1
4.4,3.0,1.3,0.2,2
5.1,3.4,1.5,0.2,0
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,1
4.4,3.2,1.3,0.2,2
5.0,3.5,1.6,0.6,0
5.1,3.8,1.9,0.4,0
4.8,3.0,1.4,0.3,1
5.1,3.8,1.6,0.2,0
4.6,3.2,1.4,0.2,2
5.3,3.7,1.5,0.2,0
5.0,3.3,1.4,0.2,0
7.0,3.2,4.7,1.4,0
6.4,3.2,4.5,1.5,0
```

Figure 4.1-3 IRIS DATA NOUVEAU_MINKOWSKI

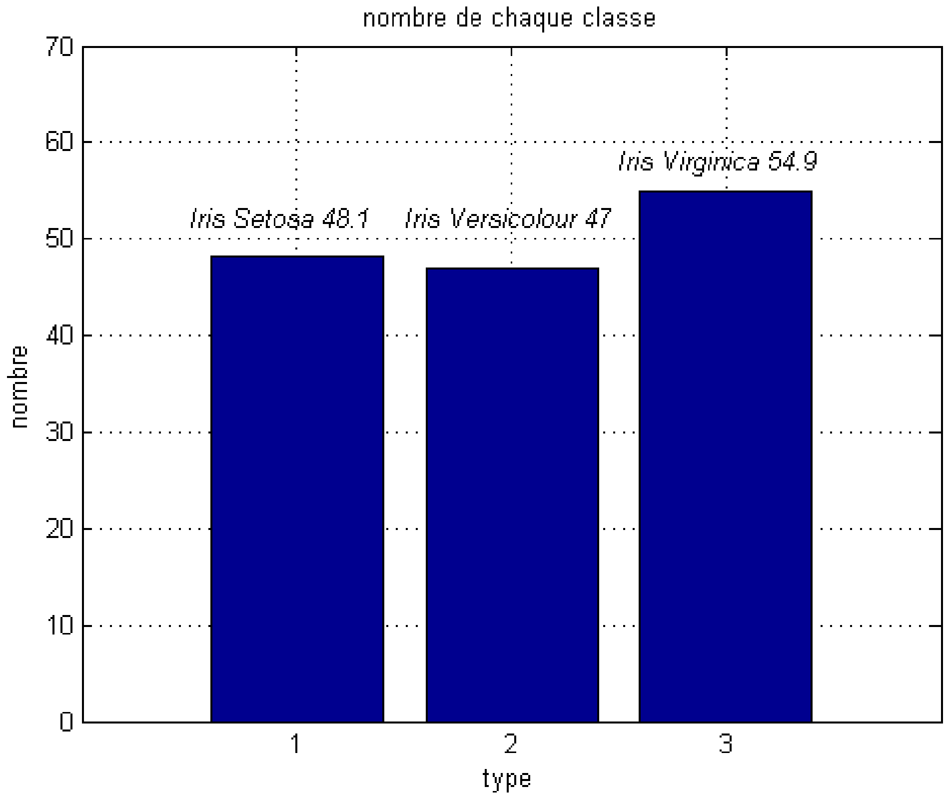

Figure 4.1-3 Pourcentage de classe distribution

De plus, on a aussi testé le pourcentage de classe distribution qui inclut 150 nombres de points et 4 attributs des points, et le but est de vérifier si l'on trouve ce pourcentage est 33.3% pour chaque classe. On a donc fait au moins de 10 fois exécution et on a calculé les valeurs moyennes de chaque classe.

Selon le schéma ci-dessus, on peut constater que le pourcentage pour chaque classe est presque équilibré, c'est-à-dire, chaque classe occupe 33.3%.

4.2 TEST SUR L'ALGORITHME K-MEANS

On a testé mon programme sur différent base de données avec les choix différents que l'on fait.

On a réalisé 2 choix pour K, 2 choix pour générer des centres, 3 choix pour la dsitance. Ci-dessous on seulement présente deux cas.

Pour le premier schéma, on utilise k=3 et la distance Euclidienne, on génère la data 'random' qui inclut la ligne 100 et la colonne 5. Après exécuter le programme, on peut constater des centres originaux et des nouveaux centres. De plus, le temps d'exécuter ce programme est 0.125486.

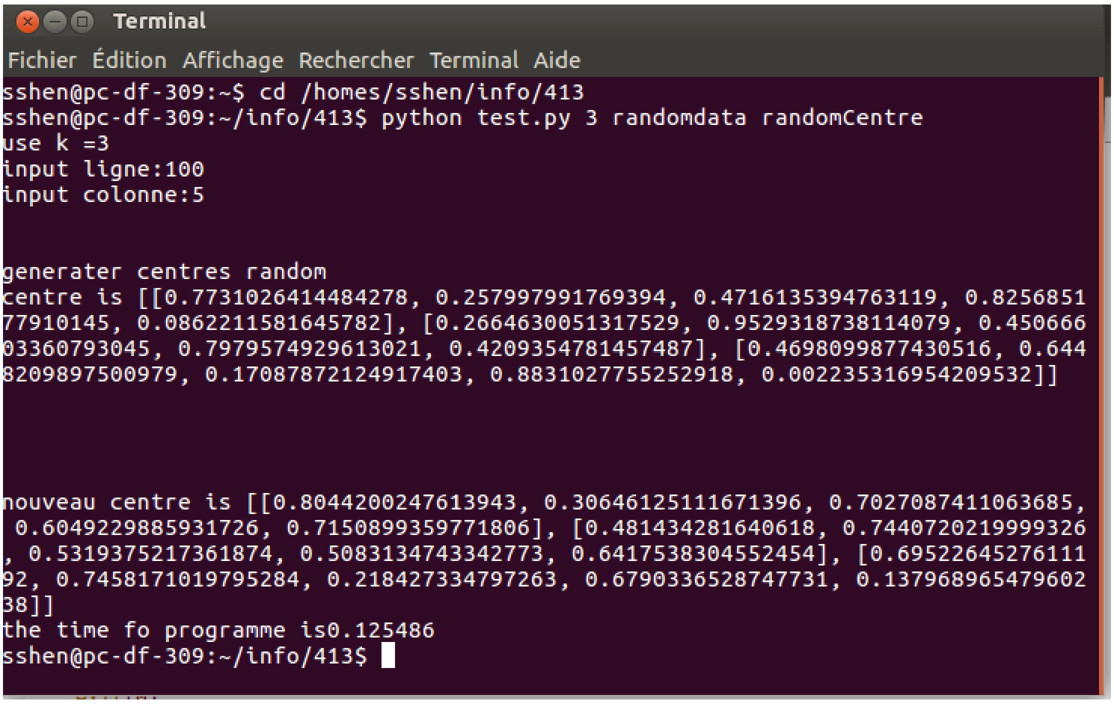

Figure 4.2-1 GÉNÉRER LA DATA RANDOM

Pour le deuxième schéma, on utilise aussi k=3, la distance Euclidienne et la data IRIS. Après exécuter le programme, on peut constater des centres originaux et des nouveaux centres. De plus, le temps d'exécuter ce programme est 0.008894.

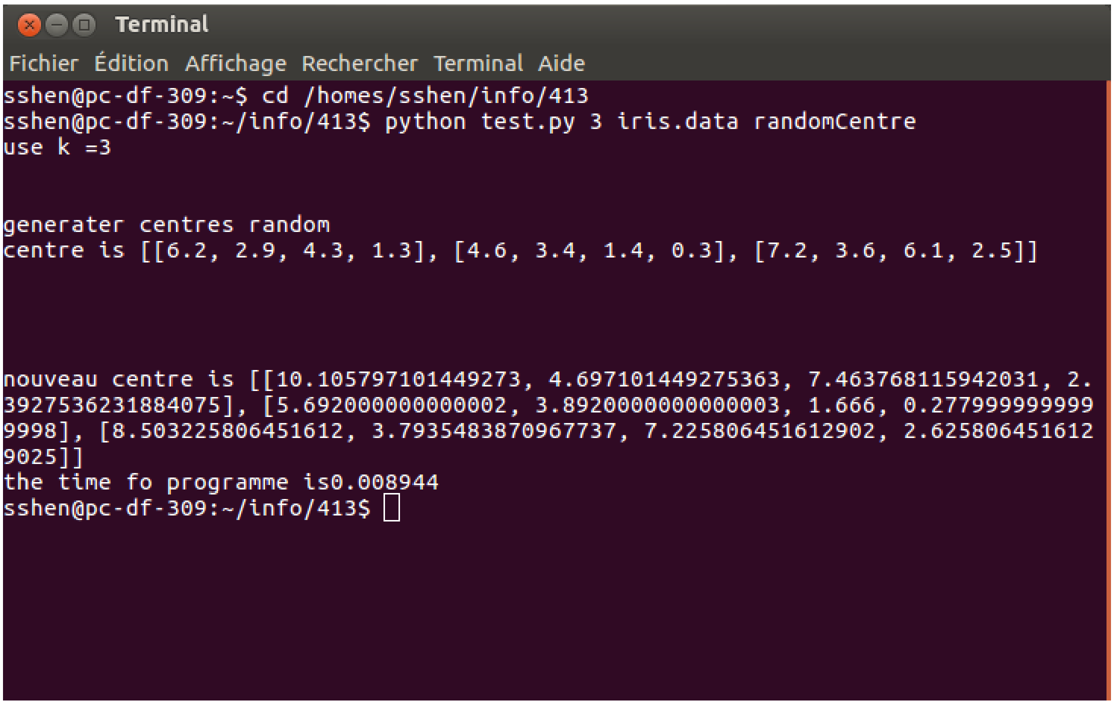

Figure 4.2-2 LA DATA IRIS

4.3 QUALITE DU CLUSTERING

Pour tester qualité du clustering, on a déjà utilisé la fonction ***matplotlib*** pour regarder les partitionnements des donnes. Ces attributs sont uniformément répartis entre 0 et 1 et la dimension n'égale qu'à 2.

Dans la première figure ci-dessous on a défini que k=4 et 500 points. Dans la deuxième figure on a défini que k=5 et 500 points. Dans la troisième figure on a défini que k=8 et 500 points.

On peut constater qu'inégalité en nombre de points au tout début mais après le partitionnement de données, il est alors assez bien équilibré.

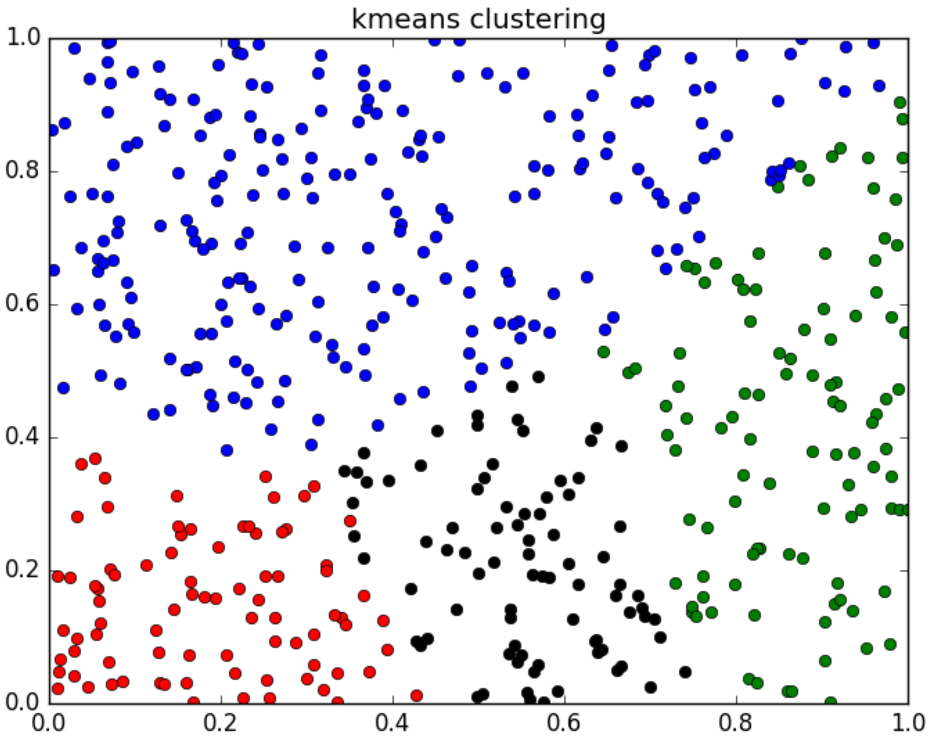

Figure 4.3-1　Clustering, k=4, point=500

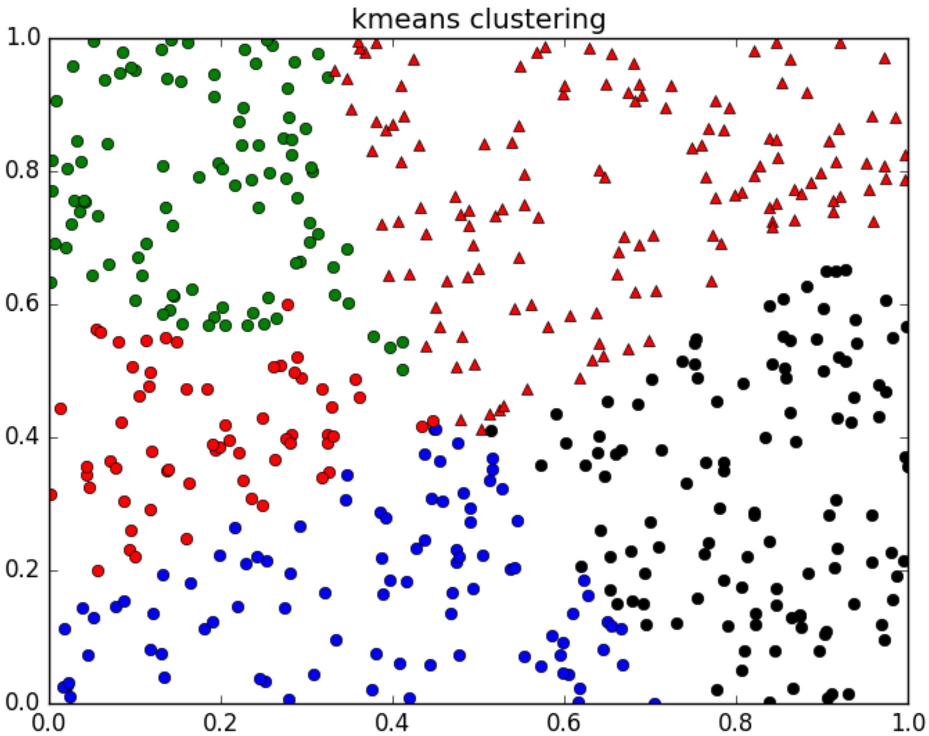

Figure 4.3-2 Clustering, k=5, point=500

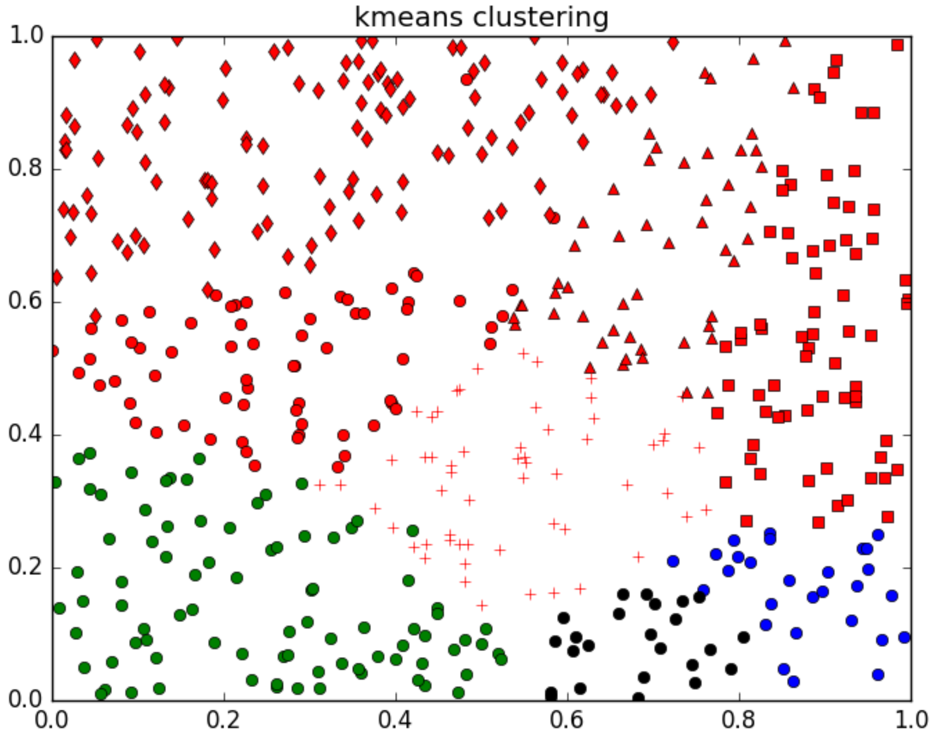

Figure 4.3-3 Clustering, k=8, point=500

5. ETUDE DE LA COMPLEXITE

Pour la complexité de l'algorithme implémenté, on a fait trois différentes variantes: le nombre d'attributs des points (entre 2 et 10), k le nombre de clusters (entre 2 et 10), et enfin le nombre de points (entre 200 et 2000).

Pour décrire ce fait, on a utilisé la fonction de temps ***time.clock()*** pour compter le temps et le logiciel ***Matlab*** pour le dessiner par appliquer la théorie d'analyse de régression.

5.1 INFLUENCE DU NOMBRE DE POINTS

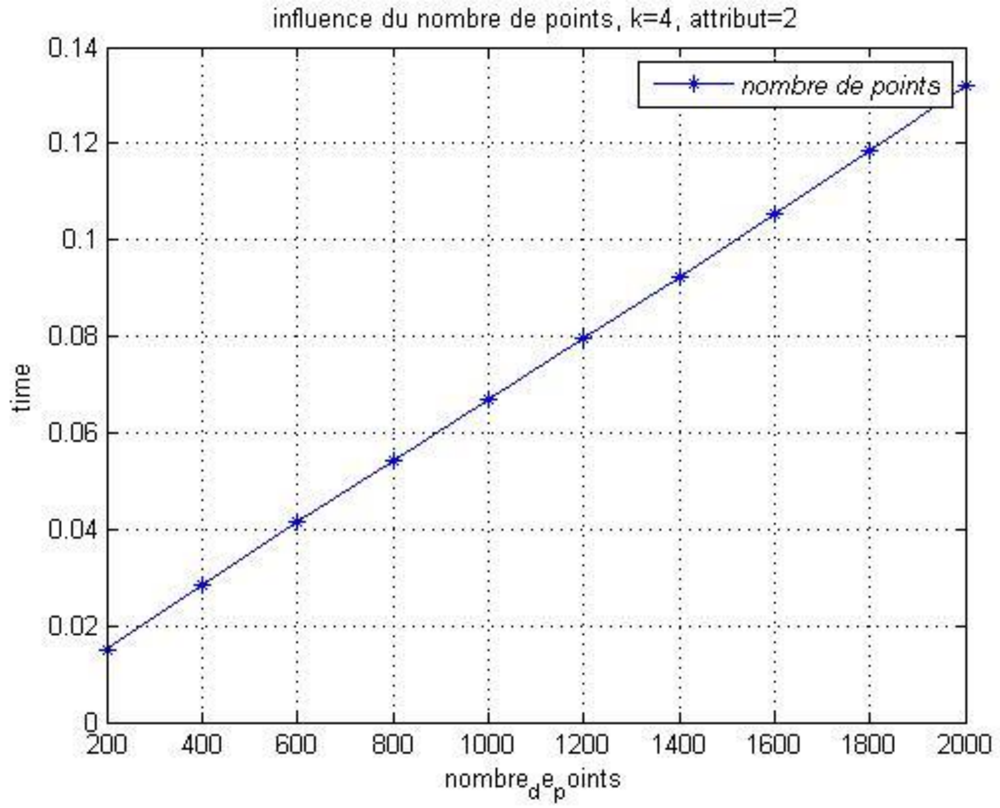

Figure 5.1

La *figure 5.1* présente le temps moyen d'exécution de l'algorithme K-means sur l'influence du nombre de points. On a défini k=4 et attributs = 2, on peut donc constater que plus le nombre de points, plus de temps d'exécution.

5.2 INFLUENCE DE CLUSTERS K

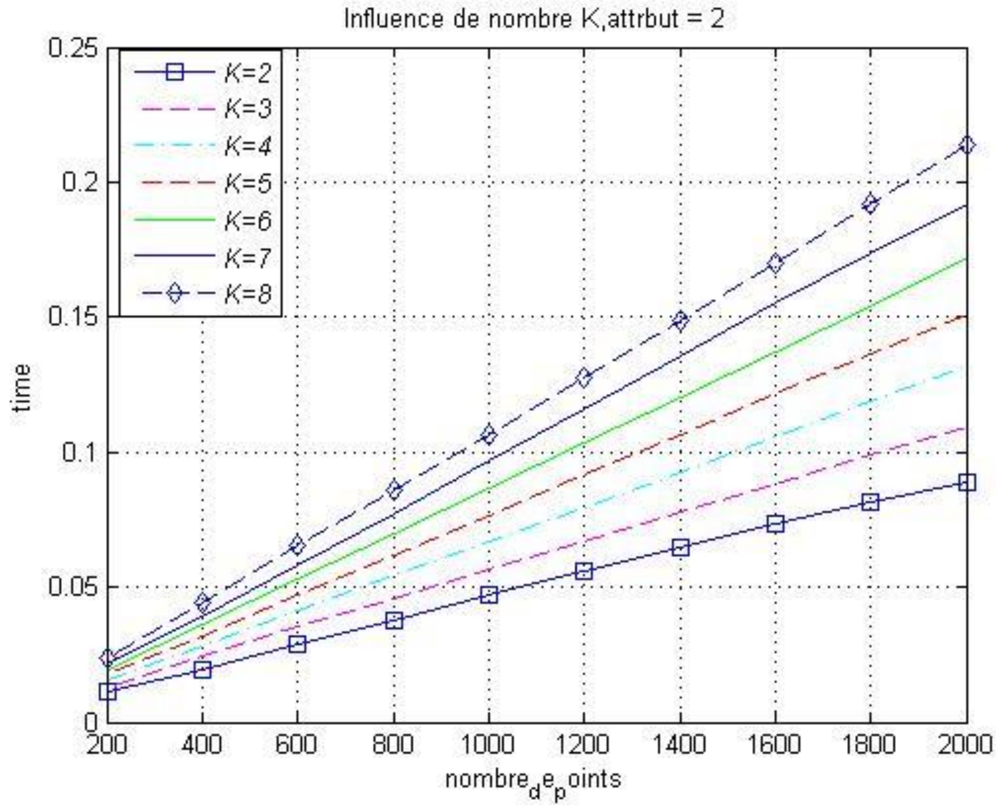

Figure 5.2

La *figure 5.2* présente le temps moyen d'exécution de l'algorithme K-means sur l'influence de nombre de clusters K. On a défini attributs = 2, on peut donc constater quand le nombre de clusters K augmente, il y a plus de temps d'exécution.

5.3 INFLUENCE DE NOMBRE D'ATTRIBUTS DES POINTS

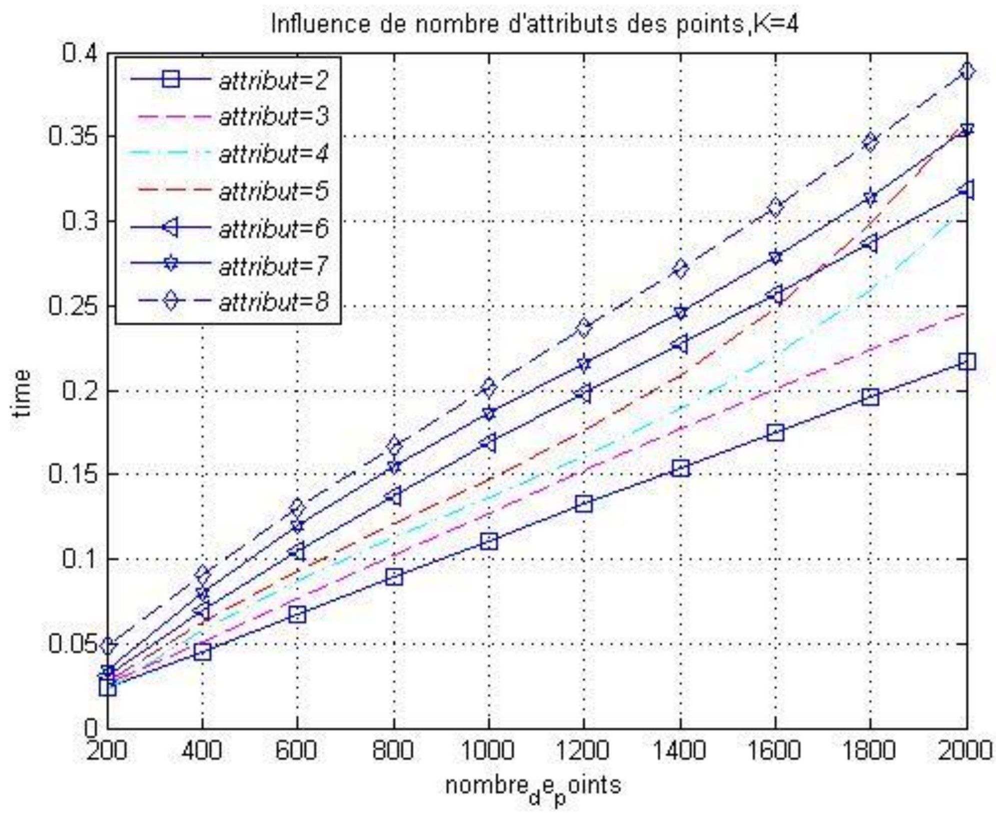

Figure 5.3

La *figure 5.3* présente le temps moyen d'exécution de l'algorithme K-means sur l'influence de nombre d'attributs des points. On a défini K = 4, on peut donc constater que plus de temps d'exécution avec l'augmentation d'attributs.

On a donc calculé la complexité de mon programme qui est de l'ordre en $O(kNid)$, avec k le nombre de clusters, N le nombre de points, i le nombre d'itération de l'algorithme jusqu'à la convergence, d le nombre d'attributs.

De plus, si k et d sont fixés, la complexité est en $O(n^{dk+1} \log n)$.

6. DISCUSSION DE L’ALGORITHME

6.1 LES DIFFICULTES LIEES A L’ALGORITHME

Pendant ce TP, on a confronté beaucoup de problèmes, surtout à utiliser l’indice de liste. Comme le code Python qui inclut l’écriture de fichiers et la génération de données aléatoires, il gère la data aléatoire avec un indice avant chaque données. Quand on veut utiliser cette data, on a confondu les indices. De plus, quand on a dessiné les schémas de partitionnement de données, on a décidé d’utiliser un tuple à l’appeler pour éviter les erreurs d’indice.

6.2 AMELIORER DE L’ALGORITHME

6.2.1 Choisir le “k”

Les algorithmes de clustering sont des outils utiles pour l'extraction de données, la compression, la probabilité estimation de densité, et beaucoup d'autres tâches importantes. Cependant, la plupart des algorithmes de regroupement nécessitent utilisateur de spécifier le nombre de grappes (appelées k), et il ne sait pas toujours ce qui est la meilleur valeur pour k.

On présente donc un algorithme simple appelé G-means qui découvre un lieu K à l'aide d'un test statistique. L'algorithme G-means adopte une approche hiérarchique pour détecter le nombre de grappes. Il teste à plusieurs reprises si les données dans le voisinage d'un barycentre de cluster semblent gaussien, et sinon il divise le cluster. A force de G-moyens est qu'il traite bien avec des données non-sphériques (étirées clusters).

6.2.2 Choisir des centres

On peut améliorer cet algorithme K-means par choisir des mieux centres. On doit d’abord chercher une valeur minimum et une valeur maximum dans chaque ligne, et puis on peut définir un intervalle entre le minimum et le maximum. Enfin on génère les centres qui entre ce minimum et ce maximum. Ainsi on peut contrôler des mieux centres pour diminuer les distances entre les points.

6.3 CONCLUSION

On a déjà réalisé cet algorithme K-means pour résoudre le problème de partitionnement de données et on a aussi confronté beaucoup de problèmes. Il m’a aidé à améliorer la programmation Python et bien comprendre l’algorithme K-means.

7. LA BIBLIOGRAPHIE

[1] **https://fr.wikipedia.org/wiki/K-moyennes**

[2] **E. Lebarbbier, T. Mary-Huard.** ***Classification non supervisée*, chapitre 2- Méthodes de partitionnement**

[3] **https://blog.bigml.com/2015/02/24/divining-the-k-in-k-means-clustering/**

[4] **Greg Hamerly, Charles Elkan,** ***Learning the k in k-means***

www.telecom-bretagne.eu

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
Tél. : + 33 (0)2 29 00 11 11
Fax : + 33 (0)2 29 00 10 00

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
Tél. : + 33 (0)2 99 12 70 00
Fax : + 33 (0)2 99 12 70 19

Campus de Toulouse
10, avenue Edouard Belin
BP 44004
31028 Toulouse Cedex 04
France
Tél. : +33 (0)5 61 33 83 65
Fax : +33 (0)5 61 33 83 75

TELECOM Bretagne