Université Paris Dauphine

Data Analytics

Implementation de KMeans avec Spark

Etudiants
Elie Abi Hanna Daher
Bilal El Chami
Badr Erraji

Professeur
M. Benjamin Negrevergne

29 avril 2018

Table des matières

1 Travail effectué **2**
1.1 Implementation 2
1.1.1 Kmeans 2
1.1.2 Generateur 2
1.1.3 Plot 4
1.2 Simulation 5
1.2.1 Kmeans 5
1.2.2 Executions 6

2 Analyse **6**

3 Application **6**

La méthode K-means est une des méthodes de clustering les plus utilisée lors de l'implémentation d'algorithmes cherchant à grouper un ensemble de données disparates. Cette méthode repose principalement sur le *unspervised learning*. L'objectif étant alors de grouper ces dernières en 'implémenté l'algorithme KMean avec Spark (python) et en évaluant la performance de notre implémentation basé sur des données que nous générons.

1 Travail effectué

1.1 Implementation

1.1.1 Kmeans

1.1.2 Generateur

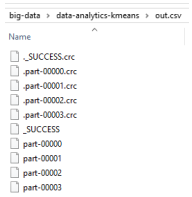
Afin d'évaluer l'algorithme KMeans implémenté, un générateur de données est mis en place.

On a mis en place 2 fichier de python : *generator.py* et *generator-noise.py*
Le fichier *generator.py* permet de générer des points et de les sauvegarder dans un fichier.
Le deuxième fichier, *generator-noise.py*, permet de générer les points ainsi qu'il génère des points bruits qui sont générer aléatoirement.

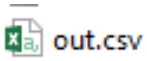
Afin de sauvegarder les données dans un fichier externe, on a implémenté 2 façons différentes :

— *saveAsTextFile* : qui est une méthode déja présente qui sauvegarde un rdd
— *write-into-csv* qui est une méthode que nous avons créer qui permet d'écrire directement dans un fichier les valeurs du RDD

La méthode que nous avons créer, *write-into-csv*, permet de generer un seul fichier mais n'est pas efficient et ne profite pas des caractéristiques du RDD c'est pour cela la méthode saveAsTextFile est meilleur en terme d'efficacité. Cette derniere créer plusieurs fichiers.



(a) Méthode *saveAsTextFile*



(b) Méthode *write-into-csv*

FIGURE 1 – Fichier output de chaque méthode du générateur de données

En exécutant, le *generator.py* avec la commande suivantes :

```
$ spark-submit generator.py out 9 3 2 10
```

Les points sont générés, et en affichant les résultats dans le graphique on obtient :

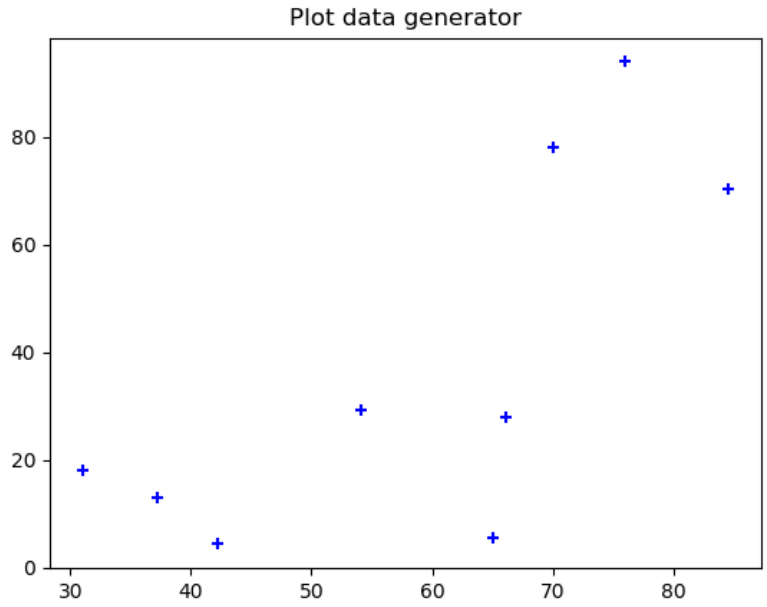


FIGURE 2 – Generateur des points

En executant, le *generator-noise.py* avec la commande suivantes :

```
$ spark-submit generator_noise.py out 9 3 2 10
```

Le nombre de bruits est le double du nombre des points à générer. Les points sont générés, et en affichant les resultats dans le graphique on obtient :

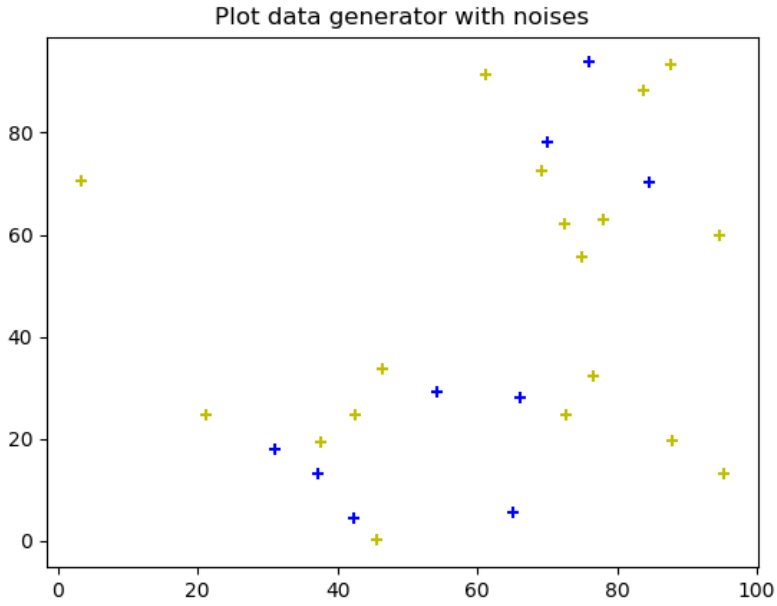


FIGURE 3 – Generateur des points avec des bruits

1.1.3 Plot

Afin de générer les graphiques, nous avons utilisés la librairies *matplotlib.pyplot*.
Pour les fichiers kmeansPlusPlus, Kmeans nous avons créer 2 fichiers pour chacun, 1 sans plot et 1 avec plot.

Pour chaque itérations, une image est sauvegarder dans le répertoire afin d'analyser l'evolution du résultat.

Les graphes générer sont de 2 dimensions, ce qui indique que si les points ont plus de 2 coordonnées ca ne seras pas tres bien representé graphiquement. Afin de voire des exemples d'execution, veuiller voir la section Simulation.

1.2 Simulation

1.2.1 Kmeans

En exécutant, le *kmeans-plot.py* avec la commande suivantes :

```
$ spark-submit kmeans-plot.py data/iris-small.dat 3 10
```

Le graphe suivant représente les résultats de la premirer iteration et en affichant les 2 premiers coordonnées : sepal width et sepal height

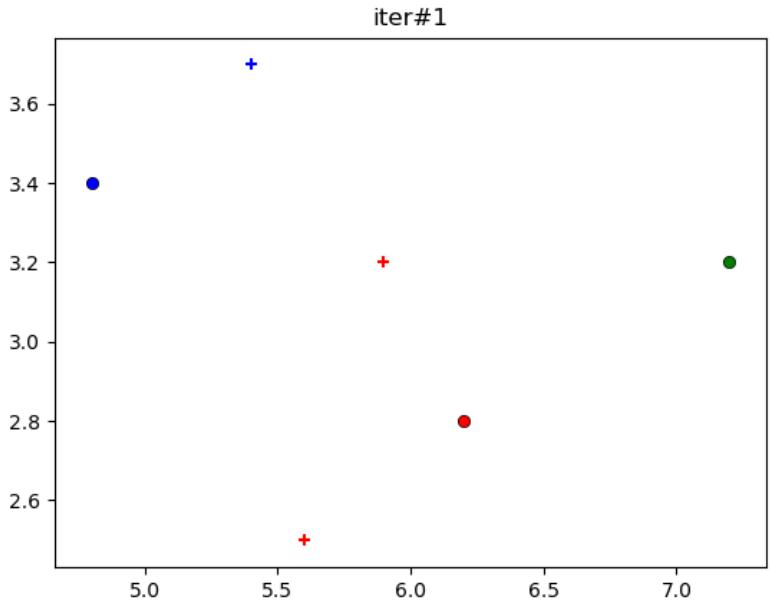


FIGURE 4 – Iteration 1

Pour trouver le resultat final on a eu besoin de 2 iterations avec une distance finale de 0.9711 d'ou le resultat final est le suivant

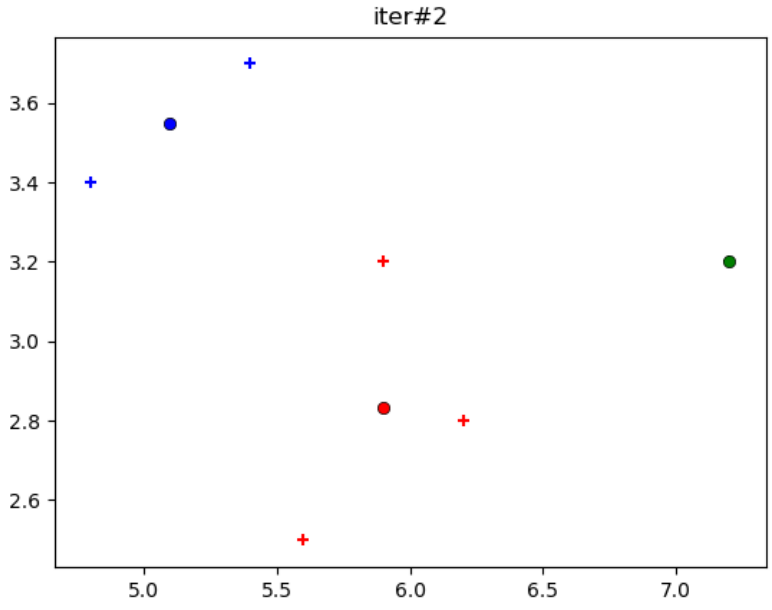


FIGURE 5 – Iteration finale

1.2.2 Executions

TABLE 1 – 100 executions

Algorithme	distance intra-cluster
KMeans - implémenté	0.02 s
KMeans++ - implémenté	0.02 s
KMeans	0.02 s

2 Analyse

3 Application

Références

[1] T. Cazenave. *Monte-Carlo Kakuro*

[2] Wikipedia page on *Kakuro*. [Online].
Available : https://en.wikipedia.org/wiki/Kakuro

[3] Wikipedia page on *CSP* . [Online].
Available : https://en.wikipedia.org/wiki/Constraint_satisfaction_problem

[4] Wikipedia page on *Backtracking algorithm*. [Online].
Available : https://en.wikipedia.org/wiki/Look-ahead_(backtracking)