K- means++

Lucas billaud

Janvier 2021

1. Introduction

Nous allons à l'aide de ce document expliquer l'étude faite par David Arthur et Sergei Vassilvitskii sur l'algorithme de clustering k-means.
Le but de leurs recherches est de proposer et de démontrer que l'on peut améliorer l'algorithme de base en k-means++.

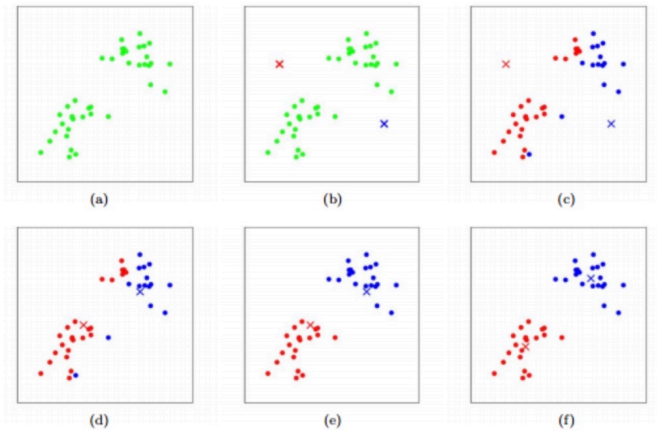
Afin d'expliquer leurs démarches et d'expliquer la différence entre les 2 algorithmes, nous allons diviser notre travail en plusieurs parties.
En premier lieu, nous nous attarderons sur k-means en donnant une explication de son utilité, son fonctionnement, son utilisation et ses faiblesses. Nous aborderons aussi dans cette partie les notions de distance et en donnant un aperçu des différentes distances qui existent. Enfin nous donnerons une définition de compétitivité au sens informatique.
Dans une deuxième partie, nous aborderons de façon large le problème qu'essaient de résoudre les auteurs. Nous détaillerons leurs raisonnements et présenterons plus en détails les étapes de k-mean et k-means++ en démontrant pourquoi k-means++ est plus précise. Nous admettrons qu'elle est d'une compléxité O(log k).
Dans cette partie nous verrons en quoi k-means++ est une amélioration de k-means. Nous mettrons en avant les avantages de k-means++ en expliquant certaines démonstrations mathématiques faites par David Arthur et Sergei Vassilvitskii. Nous ferons surtout un focus sur la façon dont les centroids sont choisis.
Enfin, dans une troisième partie, nous expliquerons les résultats obtenus par ces chercheurs en présentant un tableau de résultats empiriques.

2. Définitions autour de k-means.

La méthode k-means est une technique de machine learning non supervisée. Elle est utilisée afin de créer ce que l'on appelle des « clusters ». Le but général de cette méthode et de regrouper des points en fonction de leur « ressemblance ». Par exemple, elle pourra être utilisée sur des problématiques business de type :
- segmentation des clients : combien de types différents de clients possède mon business ?
- Optimisation du cross-selling : quels sont les produits qui ont des caractéristiques communes ?

Afin de créer ces groupes qui possèdent des caractéristiques proches, on utilise la notion de distance.
Il faut savoir qu'il existe différentes distances. Bien que nous avons l'habitude de concevoir la distance comme une longueur euclidienne qui sépare 2 points, cette longueur peut prendre plusieurs formes.
En voici quelques exemples :
- Distance euclidienne
- Distance dans les graphes : nombre d'intermédiaires entre 2 personnes
- Distance de Levenshtein : mesure de différence entre 2 mots. Nombre minimal d'action sur un mot 1 pour passer au mot 2.

Dans le cas de k-means on s'intéresse à la distance euclidienne. Le but est de minimiser la distance d'un groupe de point par rapport au centroids (point fictif choisi d'une façon aléatoire uniforme). Un groupe de points qui a le même centroid est appelé cluster. Ainsi, chaque cluster possède des caractéristiques communes.
Pour expliquer l'utilisation de la méthode k-means nous allons commenter le schéma suivant :



(a) on a un ensemble de point répartis de façon quelconque.
(b) on place 2 centroids
(c) on attribut des points à chaque centroid en fonction des distances
(d)(e)(f) on adapte l'endroit où se trouvent les centroids par rapport à l'ensemble de points en vérifiant à chaque fois que les distances soit minimisées et on répète le processus jusqu'à stabilisation.

K-means séduit en général pour sa rapidité d'implémentation dans la pratique. Elle est connue pour avoir une complexité quadratique O(n2). Afin de mieux comprendre cette notion de rapidité, voici une brève définition et un schéma pour mieux appréhender cette notion.

La complexité algorithmique (big O) est une façon de mesurer mathématiquement l'efficacité d'un code. Plus la complexité d'un code sera faible plus celui-ci sera « rapide ». Pour 2 codes qui font la même opération, il est préférable d'avoir un code avec la complexité la plus basse.

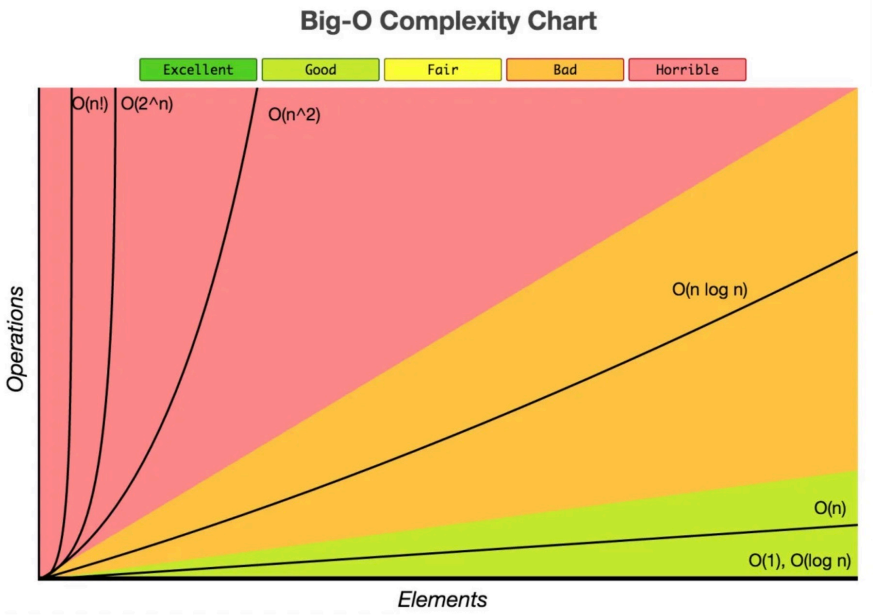
Par exemple, nous souhaitons faire une fonction qui calcule la somme des nombres de 1 à n.

```
1.  def getSum(n):
2.      sum = 0
3.
4.      for number in range(1, n + 1):
5.          sum += number
6.
7.      return sum
```

```
1.  def getSum(n):
2.      return n * (n + 1) / 2 <-- trois opérations
```

Ces 2 fonctions exécutent et retourne les mêmes résultats. Cependant, la 2nde est plus performante que la 1ère. En effet, lorsque le nombre d'input augmente la 1ère fonction doit boucler autant de fois que n est grand, elle a une complexité linéaire O(n). Quand la 2nde, peu importe la taille de n, il y a seulement une opération qui est effectuée, elle a un complexité constante O(1).

Pour résumer, la complexité d'un algorithme se base sur sa vitesse d'exécution en fonction du nombre d'input. En voici, quelques uns sur le schéma suivant.



Maintenant que les points ci-dessus ont été éclaircis, on sait que k-means fait partie des méthodes de machines learning non supervisées. Elle est basée sur la minimisation des distances (ensemble de points par rapport au centroid) elle est d'une compléxité O(n2).
Nous allons maintenant entrer un peu plus en détails dans son fonctionnement et aussi expliquer comment fonctionne k-means ++.

3. Les problèmes de k-means et une des façon de l'optimiser grâce à k-means ++

K-means -définition
Étant donnés un entier k et un ensemble de points n. Nous voulons choisir k centres C telle que la distance carrée par rapport au centroid soit minimisée.
Ainsi on définit :

.

Avec Φ la fonction qui minimise $\phi = \sum_{x \in X} \min \|x - c\|^2$ cette distance.

L'implémentation de la définition afin d'appliquer k-means se fait de la façon suivante.

1. Arbitrarily choose an initial k centers $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$.
2. For each $i \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in $\mathcal{X}$ that are closer to c_i than they are to c_j for all $j \neq i$.
3. For each $i \in \{1, \ldots, k\}$, set c_i to be the center of mass of all points in C_i: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
4. Repeat Steps 2 and 3 until $\mathcal{C}$ no longer changes.

Le problème qu'essaient de résoudre les chercheurs est d'augmenter la précision ainsi que la rapidité de k-means.

Le problème principal avec cet algorithme est le placement initial des centroids. Ces centroids sont placés de façon aléatoire et elle se stabilise au fur et à mesure du processus de stabilisation des clusters.
L'une des façons à la fois d'augmenter la précision et la rapidité de k means est décrit comme suit :

1a. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.
1b. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$.
1c. Repeat Step 1b. until we have taken k centers altogether.
2-4. Proceed as with the standard `k-means` algorithm.

We call the weighting used in Step 1b simply "D^2 weighting".

Le principe de cette amélioration est de pondérer étape par étape le choix de positionnement des centroids.
Cette amélioration apporte à la fois une meilleure précision ainsi qu'une meilleure rapidité de k-means.

Il s'en suit le théorème suivant :

Theorem 3.1. *If $\mathcal{C}$ is constructed with `k-means++`, then the corresponding potential function ϕ satisfies, $E[\phi] \leq 8(\ln k + 2)\phi_{\mathrm{OPT}}$.*

La démonstration de ce théorème étant assez longue et complexe, nous allons essayer de commenter les parties importantes.

La première partie de la démonstration consiste à montrer que les clusters formés avec k-means ++ sont plus précis.
Elle s'appuie sur l'équation suivante, qui montre le positionnement d'un point arbitraire dans un cluster par rapport au centroid.

$$\sum_{x \in S} \|x - 2\|^2 - \sum_{x \in S} \|x - c(s)\|^2 = |S| \cdot \|c(s) - 2\|^2$$

Cette équation permet de démontrer que

$$E[\phi(A)] = 2 \sum \|a - c(A)\|^2$$

Cad que cela correspond à un cluster optimal.

$$E\left[\phi \quad (A)\right] \leq \frac{2}{A} \cdot \sum_{a_0 \in A} \frac{\sum_{a \in A} D(a)^2}{\sum_{a \in A} D(a)^2} \cdot \sum \min\left(D(a), \|a - a_0\|\right)^2) +$$
$$\frac{2}{A} \cdot \sum_{a_0 \in A} \frac{\Sigma_{a \in A} \|q - a_0\|^2}{\Sigma_{a \in A} D(a)^2} \cdot \sum \min\left(D(a), \|a - a_0\|\right)^2)$$

Il faut ensuite montrer que choisir les centroids en utilisant « D2 weighting » est plus efficace.
Les auteurs démontrent que cela permet de mieux positionner les centroids au fur et à mesure.

	Average ϕ		Minimum ϕ		Average T	
k	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	10898	5.122	2526.9	5.122	0.48	0.05
25	787.992	4.46809	4.40205	4.41158	1.34	1.59
50	3.47662	3.35897	3.40053	3.26072	2.67	2.84

La seconde consiste à montrer que k-means ++ est de complexité O(log).

Nous ne commenterons pas cette partie. Celle-ci est trop complexe et aborde des notions mathématiques assez poussées.

4. **Résultats**

Afin de vérifier les résultats, une batterie de tests ont été menés par les auteurs. Voici un tableau qui montre leurs résultats (Se référer au document original pour en avoir l'analyse complète et les hypothèses mises en place).
Ils ont pu observer que d'une façon générale k-means++ a une meilleure performance, que ce soit au niveau du temps ou sur la qualité de la formation des clusters (Average Φ étant moins grande avec k-means++).

Sources

https://www.jesuisundev.com/en/understand-big-o-notation-in-7-minutes/

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1?gi=ff4dad9f01b9

Encyclopédia of Distance

https://math-os.com/manipulation-sommes-symbole-sigma/