Accélération de k-means par pré-calcul dynamique d'agrégats

Nabil El Malki[*,**], Franck Ravat [*], Olivier Teste [*]

[*] IRIT (CNRS/UMR5505)
[**]Capgemini (www.capgemini.com), Toulouse, France
prenom.nom@irit.fr, prenom.nom@capgemini.com

Résumé. L'algorithme de classification non supervisé 'k-means' nécessite un accès itératif et répétitif aux données allant jusqu'à effectuer plusieurs fois le même calcul sur les mêmes données. Ces calculs répétés peuvent s'avérer coûteux lorsqu'il s'agit de classifier des données massives. Nous proposons d'étendre l'algorithme de k-means en introduisant une approche d'optimisation basée sur le pré-calcul dynamique d'agrégats pouvant ensuite être réutilisés afin d'éviter des calculs redondants.

1 Introduction

Dans le cadre des approches non supervisées, un algorithme de classification tente de diviser les données en plusieurs classes de sorte à ce que les données (appelés également individus, observations ou points) qui se trouvent dans la même classe soient les plus similaires possibles, et inversement, les points appartenant à des classes différentes soient les plus dissemblables possibles.

Parmi les algorithmes de classification, l'algorithme de k-means (centres fixes) est probablement un des plus connus (Forgy, 1965) et constitue l'objet de notre étude. Sa première version est apparue dès les années 50 (Jain, 2010). Ce dernier, repose sur un traitement itératif (i.e. les instructions de l'algorithme doivent être réalisées plusieurs fois avant de converger vers une classification stable) et répétitif (i.e. un même calcul est potentiellement effectué plusieurs fois sur les mêmes données). Dans 'k-means', les résultats d'une itération ne sont pas conservés pour alimenter l'itération suivante. Cette caractéristique engendre des dégradations de performances notamment lorsque la dimension (nombre d'attributs) est important et lorsque ces dimensions possèdent de nombreuses valeurs de densité variable.

Contribution : Afin de permettre à k-means d'offrir de meilleures performances notamment sur de gros volumes de données, nous proposons une nouvelle version de k-means basée sur un principe de pré-agrégats. Cette extension repose sur les principes sont les suivants : (i) pré-calculer et stocker les différents calculs effectués lors des itérations successives, (ii) réutiliser les pré-calculs stockés pour accélérer les itérations futures. En section 2, nous discutons l'état de l'art. Ensuite, nous exposons notre modèle (section 3) et nos expérimentations (section 4).

2 Etat de l'art

Approches k-means. L'utilisation de la version standard de k-means nécessite un temps d'exécution proportionnel au produit du nombre de classes et du nombre de points par itération. Ce temps d'exécution total est relativement coûteux en termes de calcul, en particulier pour les grands ensembles de données (Alsabti, 1997). Plusieurs extensions de la version standard du k-means ont été proposées pour accélérer les temps d'exécution (Hung et al., 2005) :

— accélération de l'algorithme par la parallélisation et la distribution des données et des traitements. Cette solution est basée sur les paradigmes MapReduce ou MPI (Zhang et al., 2013) (Zhao, Weizhong ; Ma, Huifang ; He, 2009) ;
— accélération par réduction du nombre de calculs à effectuer pour chaque itération (propriétés de l'inégalité triangulaire) (Elkan, 2003) (Hamerly, 2010) ;
— accélération par organisation ou structuration des données (Hung et al., 2005).

Approches de pré-agrégations. Les précédentes optimisations n'utilisent pas le concept de pré-calcul d'agrégats déjà été utilisé dans plusieurs domaines.

— La structuration multidimensionnelle permet d'anticiper les calculs analytiques qui sont donc pré-calculés et stockés dans des cubes de données (Gray, 1996). Dans (Deshpande, 1998), les auteurs partitionnent les données en blocs uniformes (« chunks ») qui sont réutilisés lors des calculs suivants.
— Dans le cadre des données statistiques, des travaux intégrant le pré-calcul des agrégats ont été proposés par Wasay et al. (2017). Dans ce système baptisé « Datacanopy », un arbre binaire contenant des pré-agrégats est défini. Les pré-agrégats des nœuds fils sont imbriquées dans ceux des pères.

Tous ces travaux reposent sur des calculs d'agrégats statiques. Or, dans le cadre de l'apprentissage automatique, il n'est pas possible d'anticiper les calculs à pré-agréger. Notre approche repose donc sur un principe de stockage « à chaud » des pré-agrégats.

3 Contribution

3.1 Modèle

Notations de base. Soit $A = \{a_i | i = 1, \ldots, n\}$ un ensemble d'attributs d'un vecteur n-dimensionnel et $X = \{x_i | i = 1, \ldots, r\}$ un ensemble de données correspondants aux instances de A. L'algorithme de k-means sépare X en K classes d'une parition $C = \{C_j | j = 1, \ldots, K\}$. Chaque classe a un centroïde $\{G_j | j = 1, \ldots, K\}$ tel que G_j est la moyenne des valeurs des données de la classe C_j.

Les centroïdes intermédiaires. Soient $G^{t-1} = \{G_j^{t-1} | j = 1, \ldots, K\}$ et $G^t = \{G_j^t | j = 1, \ldots, K\}$ deux ensembles des centroïdes des K classes. G^{t-1} et G^t représentent les centroïdes des classes respectivement aux itérations t-1 et t.

Les classes des identifiants des observations. Soit $CID = \{CID_j | j = 1, \ldots, K\}$ l'ensemble des indices des classes. Les différentes classes des CID et C se réfèrent aux mêmes données. Un indice CID_j est formé à partir des indices des données de la classe C_j lui correspondant.

Les agrégats. Soit $M = \{M_{cle}\}$ un ensemble d'agrégats correspondants à tous les centroïdes des classes utilisés dans le processus de k-means. A chaque agrégat est associé un indice de

CID (clé) permettant de l'identifier. M est vide au début du processus et il est alimenté au fur et à mesure que de nouveaux agrégats sont calculés.

3.2 Algorithme de k-means étendu

Dans cette section nous présentons l'algorithme de k-means étendu (voir Fig. 1) :

— Ligne 1 : La fonction initialiser() assigne des valeurs aux K centroïdes de gravité soit de manière aléatoire soit via une méthode d'initialisation comme dans le cas de k-means++ (Arthur et Vassilvitskii, 2007).
— Ligne 7 : $argmin|x - G_j^{t-1}|$ renvoie l'indice j du centroïde le plus proche de x.
— Ligne 13 : la fonction trier(CID_j) trie les indices des données contenus dans CID_j dans l'ordre croissant.
— Lignes 14-15 : concatener(CID_j) concatène les indices des données contenus dans CID_j. Le symbole «-» est placé entre chaque paire d'indices. Exemple : 4-7-9. Le résultat de la concaténation est assigné à l'indice «cle» . Ce dernier permet d'identifier l'agrégat M_{cle}.
— Ligne 17 : Si les données sont de dimension d alors moyenne(C_j) renvoie la moyenne des valeurs des données de la classe C_j pour chaque dimension. Le résultat est un vecteur de moyennes de taille d.

```
   input : X, K ≤ |X|
   output : C
1  G^{t-1} ⟵ initialiser() ;
2  converge ⟵ Faux ;
3  tant que converge = Faux faire
4  |  G^t ⟵ ∅; C ⟵ ∅; CID ⟵ ∅ ;
5  |  /*affectation de chaque donnée au centroïde le plus proche*/;
6  |  pour chaque x ∈ X faire
7  |  |  k ⟵ argmin|x - G_j^{t-1}|; C_k ⟵ C_k ∪ {x}; CID_k ⟵ CID_k ∪ {indice(x)};
8  |  fin
9  |  /*calcul du centroïde de chaque classe*/;
10 |  pour j allant de 1 à K faire
11 |  |  CID_j ⟵ trier(CID_j); cle ⟵ concatener(CID_j) ;
12 |  |  si ∃ M_cle alors
13 |  |  |  G_j^t ⟵ M_cle ;
14 |  |  sinon
15 |  |  |  M_cle ⟵ moyenne(C_j); G_j^t ⟵ M_cle ;
16 |  |  fin
17 |  fin
18 |  si G^t = G^{t-1} alors
19 |  |  converge ⟵ Vrai ;
20 |  sinon
21 |  |  G^{t-1} ⟵ G^t;
22 |  fin
23 fin
```

FIG. 1: Pseudo-code de l'algorithme de k-means étendu

4 Expérimentations

Pour valider notre approche, nous utilisons la plate-forme de calcul OSIRIM de l'IRIT (cluster de 24 nœuds ayant chacun 8 processeurs de 7.5 Go de mémoire). Nous utilisons un jeu de données synthétiques composé de deux ensembles : données sphériques (DS) i.e. plusieurs groupes de données homogènes (gaussienne isotropique) et des données homogènes (DH), i.e.

un seul groupe compact de données (mélange de gaussiennes) où chaque donnée représente des valeurs réelles comprises dans l'intervalle $[-10; 10]$.

Protocole expérimental Pour chaque expérimentation, 4 paramètres sont renseignés : d la dimension de la donnée $\in [1;97000]$, t le nombre de données à générer $\in [2000;202000]$, k le nombre de classes $\in [4;20]$ et D le type de distribution (sphérique, homogène). Chaque expérimentation est exécutée 10 fois. Quelque soit la version de k-means (standard ou étendue) et nous avons toujours le même partitionnement dû à la même initialisation.

4.1 Différence de temps d'exécution entre les deux k-means avec des données sphériques et homogènes

Dans cette section, nous évaluons, pour les données sphériques et homogènes, la différence de temps d'exécution (qu'on appellera DFEMS) entre les deux versions de k-means. Elle est calculé en faisant la différence entre le temps pris par le k-means étendu et le k-means standard ; il est positif lorsque k-means étendu est favorable par rapport à k-means standard (cf tab.1).

Description des données	Nombre d'expérimentations	Cas favorables à k-means étendu	Cas favorables DFEMS >= 100s	Temps min&max (DFEMS en min)
DS	1987	1353	497	-26 à 29
DS (d <= 2000)	1155	542	39	-26 à 9
DS (d >= 2000)	832	811	458	-2 à 29
DH	774	348	51	-77 à 26
DH (d <= 2000)	313	39	2	-45 à 4
DH (d >= 2000)	461	309	49	-77 à 26

TAB. 1: Résultats des exécutions de k-means sur les données sphériques et homogènes

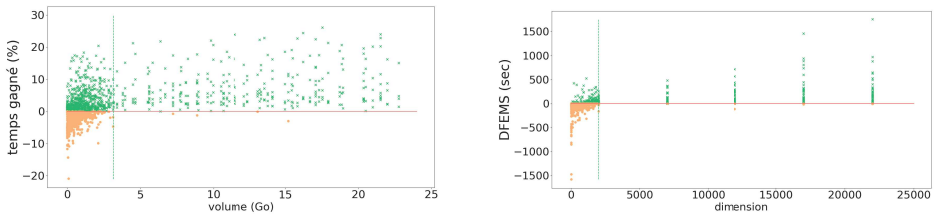

FIG. 2: k-means appliqué sur les DS ($d \in [1; 97000]$)

La version étendue présente de meilleurs résultats en termes de temps d'exécution : sur 1987 expériences sur des données sphériques toutes dimensions confondues, la version étendue présente 1353 cas favorables (68%) où le DFEMS atteint 1758s. Nous constatons également que la performance de la version étendue dépasse de plus de 100 secondes la version standard dans 25% des cas. Le nombre de cas favorables est encore plus important lorsqu'il

s'agit de données sphériques avec des dimensions d >= 2000, où on atteint un taux de cas favorables de 97% et avec 55% présentant un DFEMS supérieur à 100s. Les figures (Fig. 2) montre une ligne horizontale en rouge séparant les cas favorables (croix vertes) des cas défavorables (cercles oranges) à k-means étendu. A gauche de la figure on montre le pourcentage en gain de temps d'exécution de la version étendue par rapport au k-means standard : à partir de 3.15 Go de données, le k-means étendu est pratiquement toujours favorable allant jusqu'à jusqu'à 30% plus vite. Dans la figure 3, on observe que la version étendue de k-means est très favorable à partir de données de dimension supérieure à 2000 (ligne verticale verte) avec une DFEMS allant jusqu'à 1758s (97% de gain) alors que le temps moyen d'exécution de k-means standard est de 3473s. Les cas défavorables se concentrent principalement dans le sous-espace de dimension inférieure à 2000.

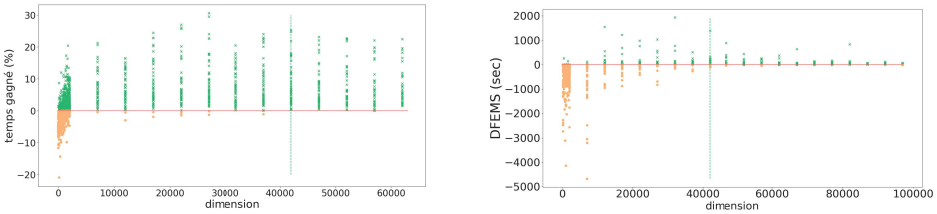

FIG. 3: k-means appliqué sur les DH ($d \in [1; 97000]$)

Dans la figure (Fig. 3), nous évaluons le comportement des deux algorithmes avec des données homogènes. Dans la partie gauche de la figure, le pourcentage de gain en temps par rapport au temps d'exécution de k-means standard est d'environ de 0 à 21% lorsque la dimension des données homogènes est égale ou supérieure à 42000. La même allure est observée dans la partie droite de la figure, la version étendue est avantageuse à partir de la dimension 42000. A partir de cette dimension, le temps moyen d'exécution de k-means étendu est de 1755 secondes. Le gain pouvant atteindre les 372 secondes.

L'approche est beaucoup plus avantageuse sur les DS que les DH puisque dans k-means appliqué à ces derniers beaucoup plus de centroïdes sont utilisés et peu de centroïdes réutilisés comparés au k-means appliqué sur les DS.

5 Conclusion

Dans cet article, nous avons proposé une approche pour optimiser le calcul de l'algorithme d'apprentissage non supervisé de données, k-means. Cette approche repose sur un algorithme qui pré-calcule et stocke les résultats intermédiaires, appelés pré-agrégats dynamiques, pour être réutilisés dans les itérations suivantes.

Nos expérimentations comparent notre version de k-means étendue avec la version standard. Notre approche montre un gain allant jusqu'à 30% sur des données de grande dimension.

Plusieurs perspectives sont envisagées. Nous allons entraîner cette version étendue de k-means avec des données plus massives encore pour mieux évaluer le coût du calcul des pré-agrégats. Nous envisageons également d'élargir notre approche à d'autres algorithmes d'apprentissage automatique des données.

Références

Alsabti, K. (1997). An efficient k-means clustering algorithm. *EECS*.

Arthur, D. et S. Vassilvitskii (2007). k-means++ : the advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1025.

Deshpande, P. M. (1998). Caching multidimensional queries using chunks. In *SIGMOD '98*.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. *ICML-2003*.

Forgy, E. (1965). Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*.

Gray, J. (1996). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery 1*(1), 29–53.

Hamerly, G. (2010). Making k -means even faster. *2010 SIAM international conference on data mining (SDM 2010)*, 130–140.

Hung, M.-C., J. Wu, et J.-H. Chang (2005). An efficient k-means clustering algorithm using simple partitioning. *JISE 1177*, 1157–1177.

Jain, A. K. (2010). Data clustering : 50 years beyond K-means. *Pattern Recognition Letters 31*(8), 651–666.

Wasay, A., X. Wei, N. Dayan, et S. Idreos (2017). Data canopy. *SIGMOD '17*.

Zhang, J., G. Wu, X. Hu, S. Li, et S. Hao (2013). A parallel clustering algorithm with mpi – mkmeans. *Journal of Computers 8*(1), 10–17.

Zhao, Weizhong; Ma, Huifang; He, Q. (2009). Parallel K-Means Clustering Based on MapReduce. *Cloud Computing. Springer Berlin Heidelberg 2009*, 674–679.

Summary

The well-known unsupervised classification algorithm called 'k-means' requires iterative and repetitive access to the data, which goes to the same (detailed) data many times over. These repeated calculations can prove costly especially when it comes to classifying massive data. In this article we propose to extend the k-means algorithm by introducing an optimization approach based on the dynamic pre-calculation of aggregates that can then be reused to avoid redundant calculations.