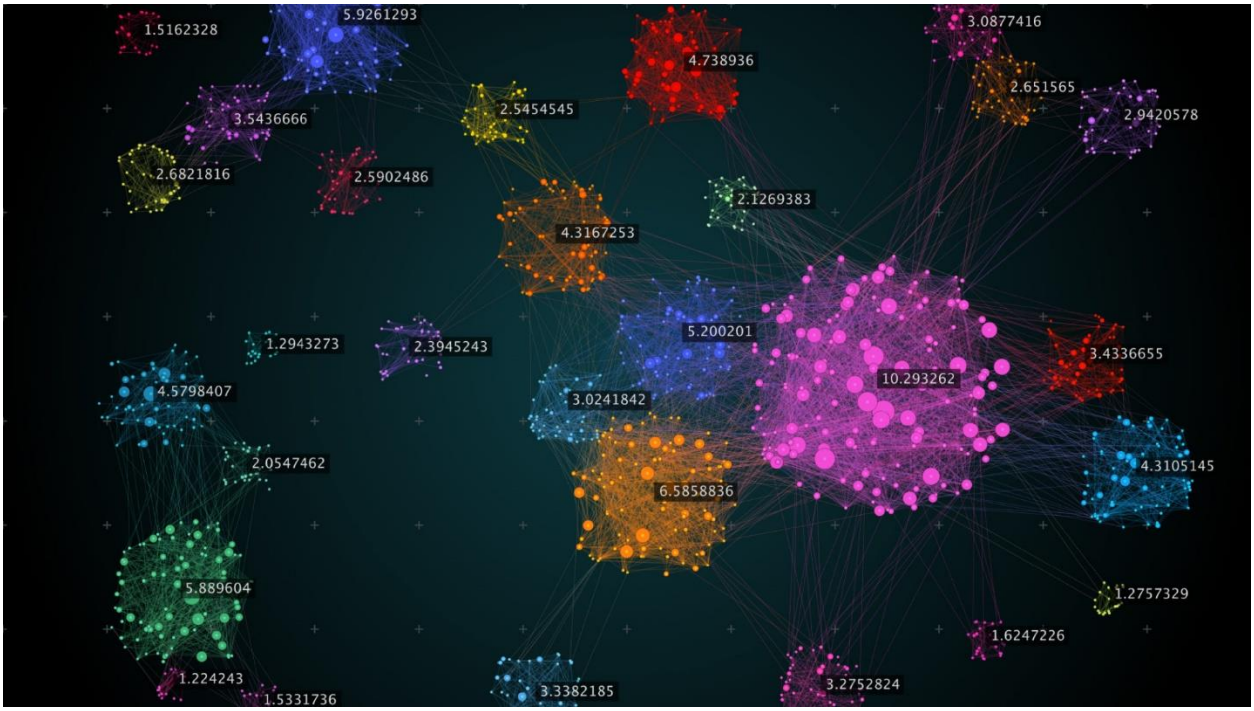
Université Euro-méditerranéenne de Fès

Algorithme Parallèle de K-means clustering



Réaliser par :

HAMDAOUI Wafae

Sous la supervision :

Pr.ABBAD Zakariae

Année Académique : 2021/2022

Introduction:

L'algorithme K-means est une approche itérative pour partitionner le jeu de données donné dans K différents sous-ensembles ou clusters, où chaque point de données appartient uniquement à un groupe. Il affecte des points de données dans un cluster tel que la somme des les distances au carré entre les points de données sont minimales. Plus la variation est faible nous avons à l'intérieur du cluster, plus nous obtenons de clusters homogènes.

Algorithme série et analyse de la complexité:

L'algorithme général que nous suivons:

1. Indiquez le nombre de clusters K

2. Initialiser K centres aléatoires des clusters à partir du jeu de données

3. Continuez à itérer les étapes suivantes jusqu'à ce que nous n'obtenions aucune modification du point de données centre**.**

    a) Rechercher la distance minimale entre chaque point de données et les centres.
    b) Affecter le point de données au centre le plus proche.
    c) Calculer le centre de chaque cluster en prenant la moyenne des données points de ce groupe.

⇨ Cette approche est connue sous le nom d'Expectation-Maximization. Où se trouve l'E-step affectation de différents points de données au cluster le plus proche à chaque iteration .Alors que M-step calcule la moyenne des points de données dans le cluster après chaque iteration.

⇨ On peut résumer ce algorithme dans le diagramme suivant:

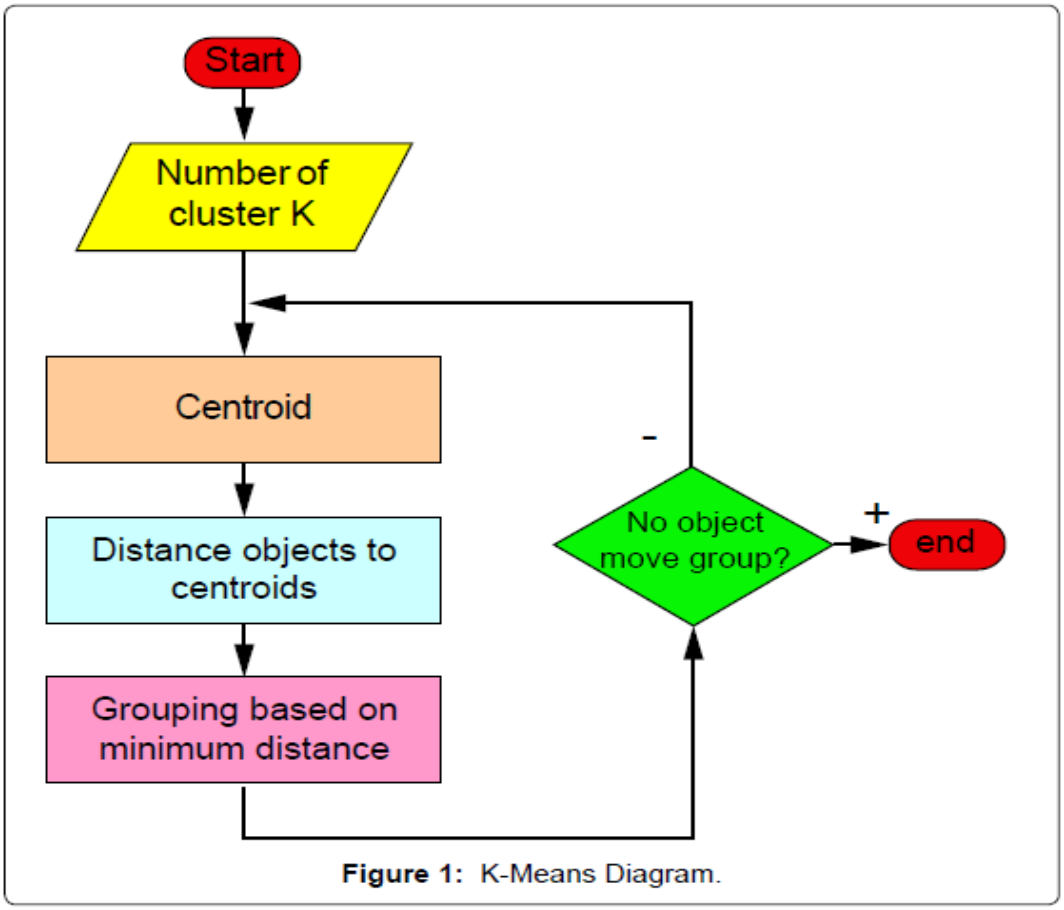


Figure 1: K-Means Diagram.

Pour voir le code cliquez cur le lien suivant :
K-means-clustering/k-means_sequential.c at main · wafaehamdaoui/K-means-clustering (github.com)

Scope de parallelism :

L'algorithme de clustering nécessite des calculs massifs, avec distance entre chaque point de données et chaque centre en cours de calcul. Depuis le calcul du centre approprié pour chaque point de données est indépendant de les autres, l'algorithme offre une bonne marge de parallélisme.

Cependant, les fils doivent communiquer entre eux-mêmes pour maintenir les valeurs centres à jour, en tant que plusieurs threads peut essayer d'accéder au même point centre. Dans ce cas, il est de s'assurer que les deux threads n'essaient pas de modifier le centre en même temps, car cela pourrait entraîner des valeurs corrompues et un faux partage.

Les Stratégies de parallélisation:

Pour la parallélisation de l'algorithme k-moyennes, un parallélisme de données a été adoptée. Les points N étaient répartis également entre les nombres de threads (en cas de répartition imparfaite des données entre threads, le reste points alloués au dernier thread)

Steps de l'algorithme :

● **Initialisation** : les premiers points de données K sont choisis comme point initial centroïdes.

● **Parallélisme des données** : chaque thread s'est vu attribuer un numéro N/num_threads Nombre de points.

● **Fonction de thread :** Chaque thread exécute une boucle (avec une valeur max_iter de 100), dans chaque itération il determine le cluster qui a le plus proche centroïde pour chaque point , puis affecte le point à ce groupe. Une fois que chaque point est affecté à un cluster, Le centre de chaque cluster est mis à jour avec les valeurs calculées à partir des points affectés à ce cluster. Il s'agit là encore d'un exemple de parallélisme des données.

● **Condition d'arrêt :** Tous les fils se cassent les itérations bouclent dès que la valeur delta passe en dessous de la Valeur seuil (1e-6).

Analyse plus approfondie de l'approche :

- ✓ **Exclusion mutuelle et section critique :** la première stratégie de synchronisation des threads qui garantit que les mises à jour du tableau des centres de cluster partagé sont entrepris par chaque thread uniquement, entrepris en utilisant la construction critique pragma omp. Ceci est essentiel pour éviter la course aux données en raison de différents threads tentant de modifier le tableau partagé.
- ✓ **Barrières** : Après chaque itération, des barrières sont utilisées pour la synchronization des thread afin que tous les threads affichent la même valeur mise à jour du tableau des centres. Deux exemples d'obstacles sont utilisés. Le premier pour la synchronisation de la mise à jour des centres et

le seconde pour synchroniser la valeur de delta, à nouveau un variable parmi tous les threads.

- ✓ Faux partage : pour minimiser le faux partage,les mises à jour des variables partagées pour le tableau des centres sont minimizes.

Vous trouvez le code parallel ici :
K-means-clustering/k-means_parallel.c at main · wafaehamdaoui/K-means-clustering (github.com)

Hardware Details:

System Model	HP Laptop 15-da1xxx
System Type	x64-based PC
System SKU	8XE85EA#BH4
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Pr...
BIOS Version/Date	Insyde F.30, 6/2/2020
SMBIOS Version	3.0
Embedded Controller Version	70.35
BIOS Mode	UEFI
BaseBoard Manufacturer	HP
BaseBoard Product	8532
BaseBoard Version	70.35
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Binding Not Possible
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.19041.1566"
User Name	DESKTOP-BIG47P7\Hello
Time Zone	Mountain Standard Time
Installed Physical Memory (RAM)	4.00 GB
Total Physical Memory	3.88 GB
Available Physical Memory	690 MB
Total Virtual Memory	9.21 GB
Available Virtual Memory	3.36 GB

Résultats et Discussions:

Nombre de clusters	Nombre de données	Nombre de thread	Temps d'execution sequenciel	Temps d'execution parallel
10	10 000	8	1.82	0.27
10	50 000	8	10.53	1.55
10	100 000	8	14.37	2.47
10	200 000	8	31.65	5.92
10	400 000	8	64.87	13.61
10	500 000	8	121.79	22.13
10	600 000	8	127.30	26.40
10	800 000	8	130.67	26.60
10	1 000 000	8	207.84	47.29

⇨ La différence entre le temps de calcul de différents jeux de données sur un nombre fixe de threads est different en raison de la répartition inégale des points pour chacun des fils respectifs.

Nombre de clusters	Nombre de données	Nombre de thread	Temps d'execution parallel
10	1 000 000	1	206.98
10	1 000 000	2	116.00
10	1 000 000	3	95.67
10	1 000 000	5	63.73
10	1 000 000	8	47.29
10	1 000 000	9	46.37
10	1 000 000	10	49.33

- ⇨ Comme on peut remarquer, le temps de calcul de l'algorithme diminue à mesure que le nombre de threads augmente tellement que pour 10 threads, le temps de calcul global pour divers jeux de données sont assez proches de zero(0.27 pour 10000 données).

Conclusion :

L'algorithme donné est grossier, en particulier pour les données volumineuses. Dans un tel cas, il peut être une bonne idée d'implémenter le code donné sur un système distribué de mémoire. En outre, MPI fournit des bibliothèques spéciales pour la saisie de données parallèles, ce qui peut être un énorme avantage en termes d'accélération.

nous sommes tenus au regroupement de points de données en 3 dimensions. Nous pouvons tenter de regrouper des points de données multidimensionnels avec mise en œuvre de la réduction de la dimensionnalité.