Parallelization of K-Means Clustering

Report

By

Vedashree Shinde -20190802044

BTech Computer Science

D.Y. Patil International University

April 18, 2022

Contents

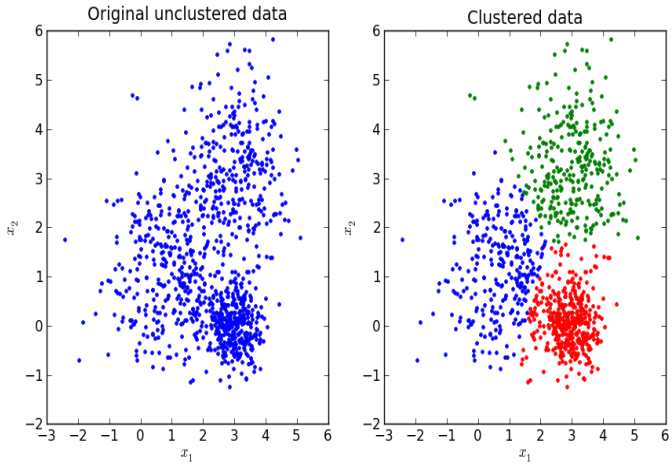
Abstract

K-means is the oldest and most popular clustering technique used, clustering techniques are the most important part of the data analysis. The K-means algorithm is a widely used partition method in clustering. It is simplest because of its easy implementation, simplicity, efficiency and empirical success.To make it more stable and portable we have tried parallelization to it. Parallelization is simply the art of performing computational work on multiple CPUs at the same time to boost performance However, the real-world applications produce huge volumes of data, thus, how to efficiently handle this data in an important mining task has been a challenging and significant issue. Therefore to overcome this problem a parallel K-means clustering algorithm with OpenMP is proposed in this paper. The algorithm enables applying the clustering algorithm effectively in the parallel environment. Experimentally the results are represented in the form of a table for time taken in Sequential and Parallel programming.

Keywords: K-means clustering, Parallelization and OpenMP

Introduction

Every learning engineer wants to get accurate predictions with their algorithms. Such learning algorithms are generally dampened into two types - supervised and unsupervised. As our paper is predicated on K-means we'll study unsupervised learning algorithms. So most unsupervised learning-based applications utilize the sub-field called Clustering. Clustering is that the process of grouping data samples together into clusters supports a specific feature that they share. A cluster refers to a group of knowledge points aggregated together thanks to certain similarities. K-means clustering is one in all the best and popular unsupervised machine learning algorithms. It's an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each datum belongs to just one group. The most objective is to get underlying patterns within the given data.In other words, k-means finds observations that share important characteristics and classifies them together into clusters. Customer Segmentation, Document Classification, House Price Estimation, and Fraud Detection are some of the important world applications of clustering.



We have implemented this algorithm i.e. K-means clustering parallelly to form it more stable and portable. So firstly lets see what's parallelization-

Parallelization:

In the simplest sense, Parallelization(parallel computing) is that the simultaneous use of multiple compute resources to resolve a computational problem:

A problem is broken into discrete parts that may be solved concurrently

Each part is further lessened to a series of instructions

Instructions from each part execute simultaneously on different processors

An overall control/coordination mechanism is used

Here for this parallelization we've got used the OpenMP interface which allows us to run programs in parallel.

Open Multi Processing (OpenMP):

OpenMP may be a set of compiler directives still as an API for programs written in C, C++, or FORTRAN that gives support for parallel programming in shared-memory environments. OpenMP identifies parallel regions as blocks of code that will run in parallel. Application developers insert compiler directives into their code at parallel regions, and these directives instruct the OpenMP run-time library to execute the region in parallel.

It creates as many threads which are processing cores within the system. Thus, for a dual-core system, two threads are created, for a quad-core system, four are created; so forth. Then all the threads simultaneously execute the parallel region.

When each thread exits the parallel region, it's terminated. OpenMP provides several additional directives for running code regions in parallel, including parallelizing loops.
In addition to providing directives for parallelization, OpenMP allows developers to settle on several levels of parallelism. E.g., they'll set the quantity of threads manually. It also allows developers to spot whether data are shared between threads or are private to a thread

Formulization

K-means clustering tries to group similar types of items within the style of clusters. It finds the similarity between the things and groups them into the clusters. K-means clustering algorithm works in three steps. So see what these three steps are.

1. We will select the k values.
2. Initialize the centroids.
3. Select the group and find the average.

Understanding each figure one by one.

- Figure 1 shows the representation of information of two different items. The primary item is shown in blue color and also the second item is shown in red color. Here I am choosing the worth of K randomly as 2. There are different methods by which we are able to choose the right k values.
- In figure 2, Join two selected points. Now to search out the centroid, we are going to draw a perpendicular line to that line. The points will move to their centroid. If you will notice there, then you may see that some of the red points are now moved to the blue points. So, these points belong to the group of blue colored items.
- The same process will continue in figure 3. we are going to join the two points and draw a perpendicular line to it and find out the centroid. So the two points will move to its centroid and again some of the red points get converted to blue points.

- The same process is happening in figure 4. These processes are going to be continued until and unless we get two completely different clusters of these groups.

NOTE: Please note that the K-means clustering uses the euclidean distance method to seek out the distance between the points.

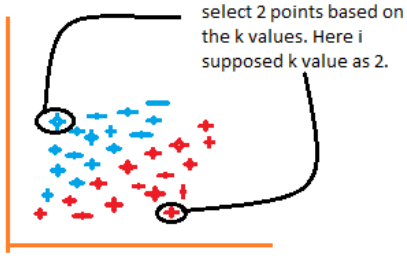
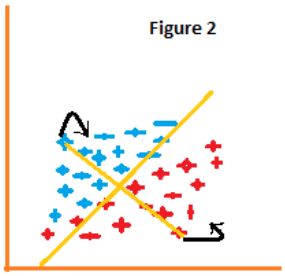


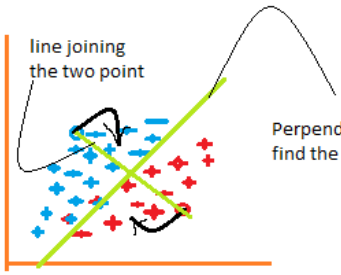
Figure 1



F2: Find the average of all the blue points and red points and move the selected points to **centroid.**

= original points

= the original points moved to centroid.



F3: Some of the red points changed to blue points, that means they belong to the group blue now. Again the repeat the same process.

F4: The same process has been applied here. This process will be continued until we get the **two complete different cluster.**

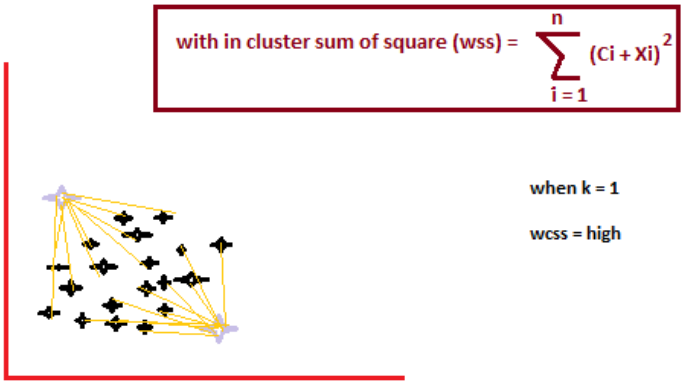
Choosing the value of K:

One of the foremost difficult tasks in this clustering algorithm is to choose the right values of k. If you are choosing the k values randomly, it'd correct or may be wrong. If you select the incorrect value then it will directly affect your model performance. So the methodology used is-

Elbow Method:

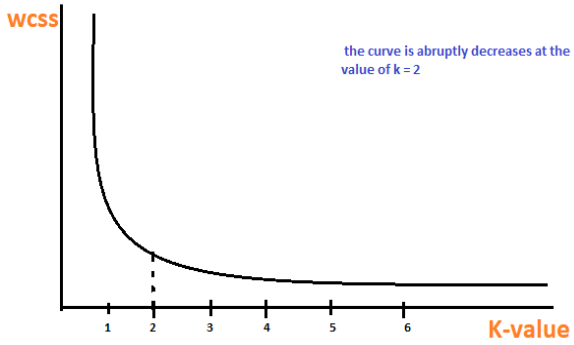
Elbow is one among the most noted strategies by which you can select the right value of k and boost your model performance. We have a tendency to also perform the hyperparameter tuning to choose the best value of k. Let us see how this elbow method works.

It associates empirical methodology to seek out the simplest price of k. It picks up the range of values and takes the best among them all. It calculates the sum of the sq. of the points and calculates the average distance.



When the value of k is 1, the sum within-cluster of the square will be high. As the value of k increases,the value of the sum of squares within the cluster decreases.

Finally, we plot a graph between k-values and the sum of the square within-cluster to get the k value. We will examine the graph carefully. At some point, our graph will decrease sharply. Therefore that point will be considered as a value of k.



This is how the K-means cluster works. Now implementation of the K-means in sequential and parallel is explained through an algorithm.

Implementation

Sequential K-means Clustering:
The basic K-means clustering technique is very simple. We start by randomly identifying K initial centroids, where K denotes the number of desired clusters and is given input by the user as an input. Next, we assign each data point to the nearest

centroid (“assignment stage”). Each resulting collection of points associated to a centroid is referred to as a cluster. We then update the centroids based on the new information from each cluster (“update stage”). We repeat the assignment and update stages iteratively until either our algorithm converges or the maximum number of iterations is reached. The steps required to performing basic K-means clustering are more formally described in Algorithm 1 below

Algorithm 1:

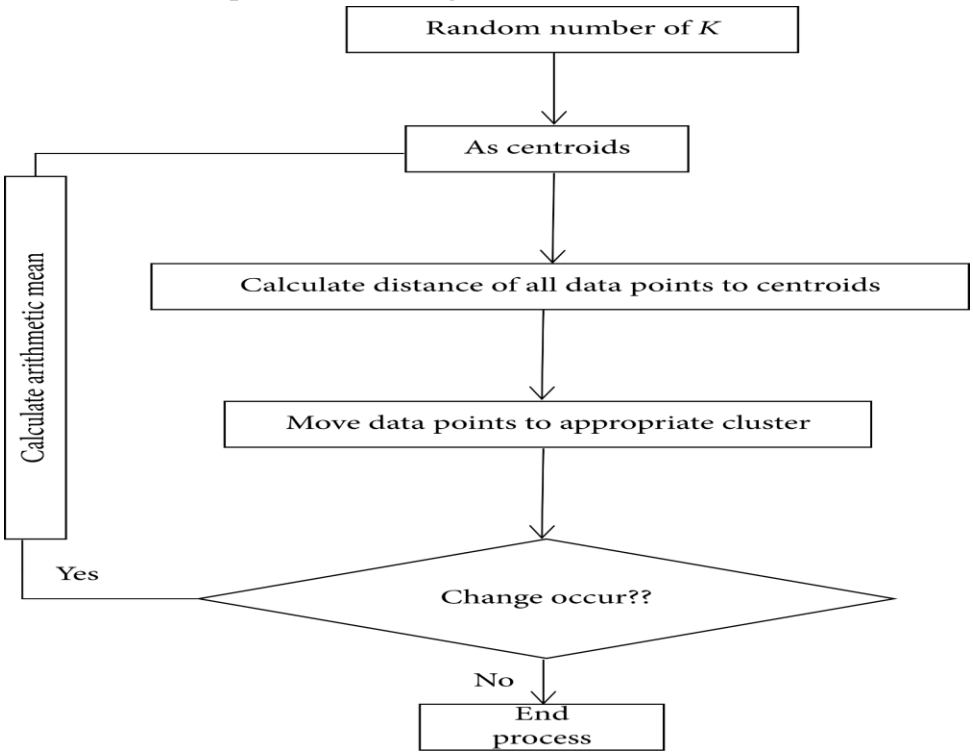
(1) Firstly randomly generate initial centroids

(2) Than, form K clusters by assigning each data point to its nearest centroid

(3) Update the centroids by calculating the cluster means

(4) Repeat steps 2 and 3 iteratively until convergence or maximum iterations is reached

- Flow chart of Simple *K*-Mean algorithm.



Parallel K-means clustering:

As our basic K-means clustering algorithm is quite simple, it is also very computationally expensive for large datasets. The algorithm requires the calculation of squared Euclidean distances between each data point and the K centroids at each iteration until convergence. In an effort to increase the computational speed of this part of the algorithm, we parallelize the process of

assigning points to clusters by dividing the task across p processors. As a result, each processor only has to complete this task for n/p data points where n is the number of observations in the dataset. Next, we aggregate the results from all p processors and use the information to update our centroids. The rest of the algorithm proceeds similarly to the basic K-means clustering algorithm. The details of our parallel K-means clustering algorithm are more formally defined in Algorithm 2 below.

Algorithm 2:

(1) First randomly generate initial centroids

(2) Than, Randomly shuffle the data points and partition them into p subgroups

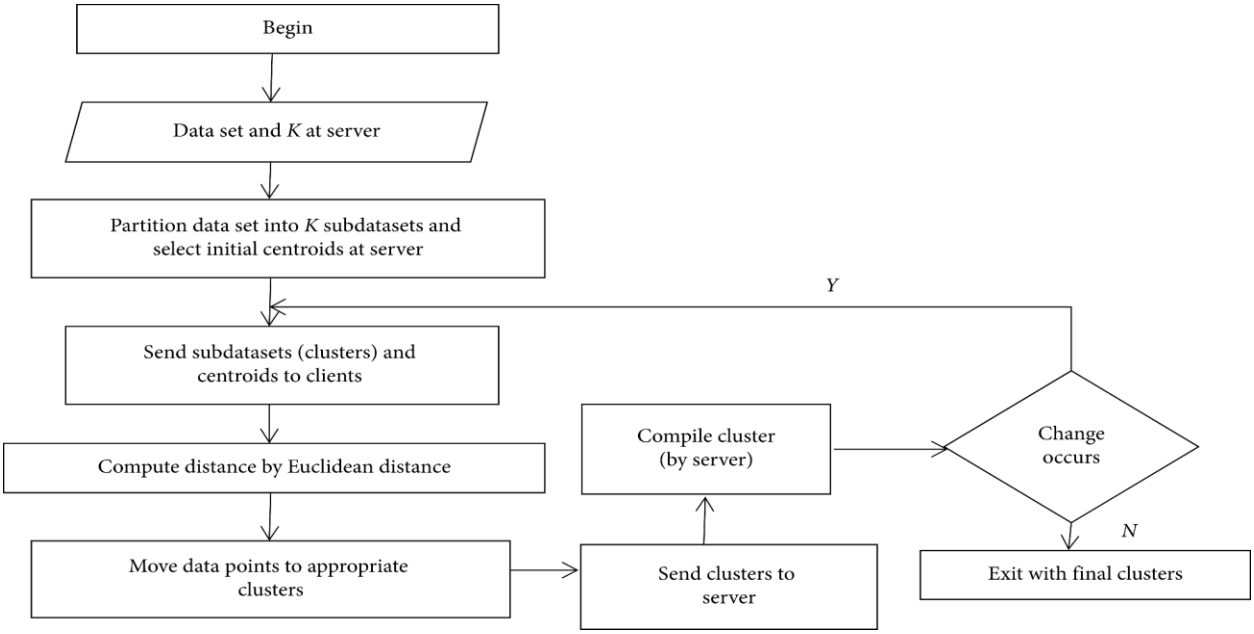
(3) Now for each subgroup p, assign each data point to its nearest centroid

(4) Aggregate/total the results from each subgroup p to form K clusters

(5) Update the centroids by calculating the cluster means

(6) Repeat steps 2-5 iteratively until convergence or maximum iterations is reached

- Flow chart of Parallel *K*-Mean clustering algorithm.



Observations and Results

The following table shows the amount of time taken by the Parallelized K Means Clustering algorithm in seconds when using different numbers of threads and when subjected to different datasets having varied numbers of data points(the readings are taken by using openmp).

N(data points)	K(no. of clusters)	Sequential	OpenMP	No. of Threads
10000	2	0.015679	0.010792	2
			0.039447	4
			0.022047	7
50000	4	0.119661	0.075805	2
			0.288157	4
			0.354774	7
100000	6	1.438202	1.782085	2
			2.091182	4
			2.722856	7
200000	7	5.073297	3.437337	2
			3.756981	4
			3.819948	7

Conclusion

Most of the existing clustering algorithms are having less efficiency due to large volumes of datasets and outliers. To achieve better accuracy with minimum time complexity, a parallel enhanced clustering algorithm is proposed on multi-core systems.This project has presented a parallelized version of the existing K-Means Clustering algorithm. In this paper K- means algorithm is implemented in both serial and parallel, the parallel method is implemented by using OpenMP. Meanwhile, the arrangement of the OpenMP parallel development environment based on C language, GCC in Ubuntu is implemented, whose ideas and procedures can be applied to Windows or other platforms. Experimental results

show that K-Means using parallel configuration is comparatively faster, stable and portable, and it is efficient in the clustering on large data sets.

References

1) https://www.hindawi.com/journals/sp/2021/9988318/
2) https://www.academia.edu/38994225/Parallelized_K-Mean_Clustering_with_ OpenMP
3) https://github.com/Saroopashree/openmp-kmeans/tree/master/datasets
4) https://www.researchgate.net/publication/319487431_Parallelization_of_K-Means_Clustering_Algorithm_for_Data_Mining