UNIVERSITÀ DI PISA

KMeans algorithm in Hadoop
Cloud Computing

ACADEMIC YEAR 2022-23

Maria Fabijan
Basma Adawy
Francesco Da Vita

Table of Contents

1. Introduction

The goal of this project was to implement the K-means clustering algorithm using the MapReduce framework. The design involved several classes, each serving a specific purpose to achieve the clustering task. The main classes included the Point, Centroid, KMeansMapper, KMeansCombiner, KMeansReducer, and the KMeans driver class.

2. MapReduce Algorithm

2.1 Point Class

The **Point** class represents a point in a multidimensional space. It has two attributes: a list of DoubleWritable values that store the coordinates of the point, and an IntWritable value that stores the number of points in the cluster that contains the point.

ToString Method

The ToString method returns a string representation of the Point object, which consists of its coordinates separated by commas.

GetCords Method

The getCords method returns the list of coordinates of the Point object.

CalculateDistance Method

The calculateDistance method takes another Point object as a parameter and returns the Euclidean distance between this Point and the other Point. This method is used to measure the similarity between two points.

Write and **ReadFields** Methods

The write and readFields methods are used to serialize and deserialize the Point object's state into a DataOutput or DataInput stream. They implement the methods from the Writable interface and write or read the size, coordinates, and cluster point count of the point.

CompareTo Method

The compareTo method takes another Centroid object as a parameter and returns 0. This method implements the method from the WritableComparable interface and is used to compare two points based on their centroids.

GetClusterPoints and **SetClusterPoints** Methods

The getClusterPoints and setClusterPoints methods are getters and setters for the cluster point count attribute of this Point. The cluster point count is an integer value that indicates how many points are assigned to the same cluster as this point.

1.2 Centroid class

The Centroid class extends the Point class, which represents a point in a multidimensional space. The Centroid class has three attributes: a list of DoubleWritable values that store the coordinates of the point, an IntWritable value that stores the number of points in the cluster associated with this Centroid, and an IntWritable value that stores the index of the Cluster.

Read and **Write** Methods
The readFields and write methods are used to serialize and deserialize the Centroid object's state into a DataInput or DataOutput stream. They override the methods from the Point class and add the index attribute to the stream.

Converged Method
The converged method takes another Centroid object and a Double threshold as parameters and returns true if the distance between this Centroid and the other Centroid is less than or equal to the threshold. This method is used to check if the centroid has converged to a stable position after an iteration of the K-means algorithm.

CompareTo Method
The compareTo method takes another Centroid object as a parameter and returns 0 if this Centroid has the same index as the other Centroid, or 1 otherwise. This method overrides the method from the WritableComparable interface and is used to compare two centroids based on their indices.

ToString Method
The toString method returns a string representation of the Centroid object, which consists of its index followed by its coordinates and cluster point count.

CalculateCentroid Method
The calculateCentroid method updates the coordinates of this Centroid by dividing each coordinate by the cluster point count. This method is used to calculate the new centroid position after assigning points to clusters.

GetIndex and **SetIndex** Methods
The getIndex and setIndex methods are getters and setters for the index attribute of this Centroid. The index is an integer value that identifies the centroid among others.

1.3 KMeansMapper Class
The mapper's task is to find the nearest centroid to each point in the input dataset and emit them as key-value pairs. The input of the mapper is a text file that contains the coordinates of the points, one point per line. The output of the mapper is a sequence of Centroid-Point pairs, where each Centroid is the closest one to the corresponding Point.
The mapper reads the centroids from a sequence file that contains the initial centroids or the updated centroids from the previous iteration.

1.4 KMeansCombiner Class
The combiner's task is to calculate the partial sum of the coordinates and the count of the points that are assigned to the same centroid by the mapper.
The input of the combiner is a sequence of Centroid-Point pairs, where each Centroid is the closest one to the corresponding Point.
The output of the combiner is a sequence of Centroid-Point pairs, where each Point represents the partial sum and the count of the points associated with the Centroid.

The combiner reduces the amount of data that needs to be transferred to the reducer and improves the performance of the algorithm.

1.5 KMeansReducer Class

The reducer's task is to calculate the new centroids by taking the average of the points that are assigned to the same centroid by the combiner.
The input of the reducer is a sequence of Centroid-Point pairs, where each Point represents the partial sum and the count of the points associated with the Centroid.
The output of the reducer is a sequence of IntWritable-Centroid pairs, where each IntWritable is the index of the Centroid and each Centroid is the updated centroid.
The reducer also writes the new centroids to a sequence file and checks if they have converged to a stable position by comparing them with the old centroids.

1.6 KMeans Class

The KMeans class is the driver class for the K-means clustering algorithm. It sets up the configuration and the job for the MapReduce framework and runs the algorithm until convergence.
Main Method
The main method takes the command-line arguments and parses them to get the number of clusters, the dimension of the points, the number of points in dataset, the input path, the output path, and the threshold for convergence.
It calls the generateRandomCentroids method to initialize the centroids and then creates a loop that runs a MapReduce job for each iteration of the K-means algorithm. It checks if the centroids have converged by looking at a counter set by the reducer and deletes the output path if not.

3. Testing

The implemented K-means clustering algorithm was evaluated using various datasets and different configurations. The algorithm demonstrated efficient clustering capabilities and convergence behavior.
The implemented K-means algorithm in Hadoop demonstrates good performance and similarity to the results obtained using the K-means algorithm from scikit-learn in Python.

3.1 Test 1

The dataset used for testing consisted of 1000 2-dimensional data points with 4 distinct clusters. For the Hadoop the threshold was set on 0.1.

- Hadoop Results:
  - Cluster 0: (4.371136957142856, 1.8467642625)
  - Cluster 1: (-6.7391497857692295, -6.915416233230769)
  - Cluster 2: (-8.85662683924303, 7.413398181035851)
  - Cluster 3: (-2.3531666336764703, 9.13857479014706)

- Python
  - Cluster 0: [4.65760511, 2.0269603]
  - Cluster 1: [-8.85662684, 7.41339818]
  - Cluster 2: [-6.74543497, -6.81351756]
  - Cluster 3: [-2.50173875, 9.03287546]

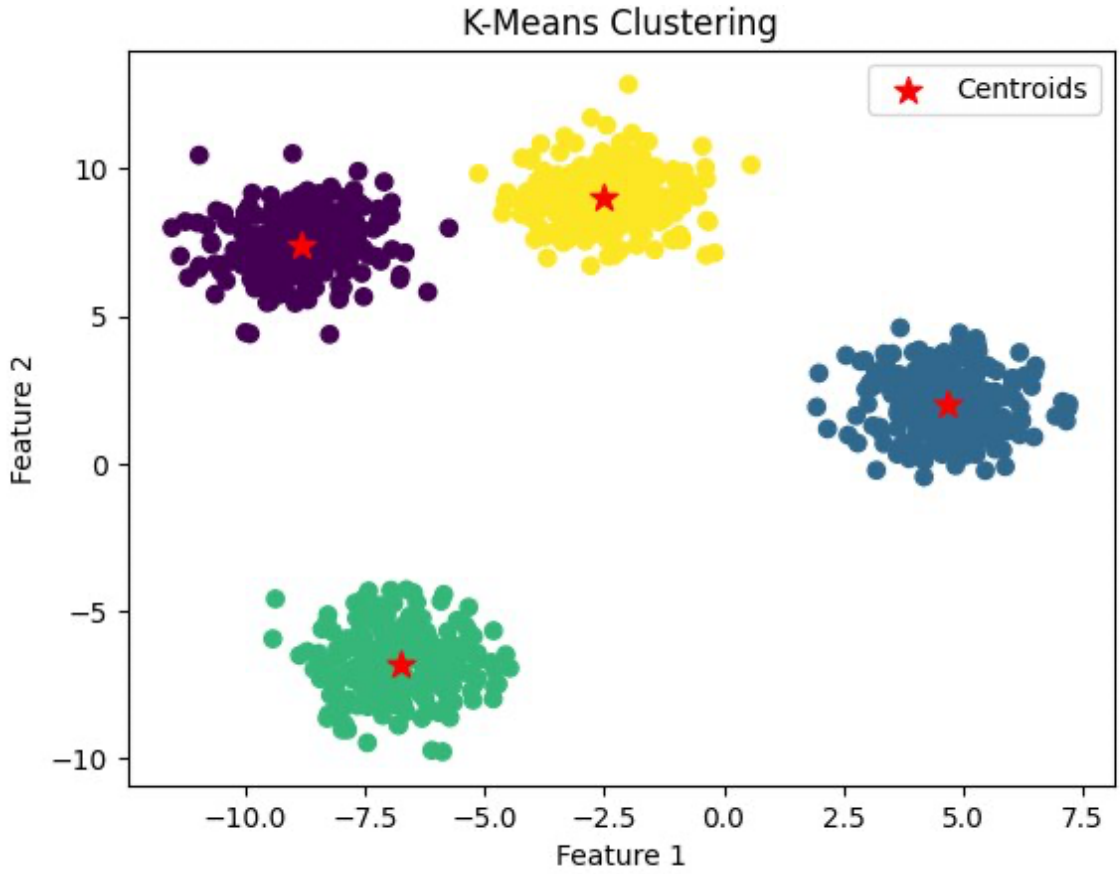

Figure 1: Dataset generated using scikit-learn library (1000 data points in 2 dimensions)

3.1 Test 2

The dataset used for testing consisted of 1000 3-dimensional data points with 5 distinct clusters. For the Hadoop the threshold was set on 1.

- Hadoop Results:
  - Cluster 0: (2.3925049971, -19.12200506804999, -17.1647322058)
  - Cluster 1: (4.271283943, -4.908983539000001, -4.238728397666666)
  - Cluster 2: (5.711182223142856, -4.349242568285715, -4.825320292000002)
  - Cluster 3: (16.032407763015073,7.658922503567836,-4.699116454924621,)
  - Cluster 4: (4.404854, -2.39547413, 6.79113271)

- Python
  - Cluster 0: [5.34161406, -4.59558932, -4.64603594]
  - Cluster 1: [2.392505, -19.12200507, -17.16473221]
  - Cluster 2: [15.29645901, -2.25476907, -2.25779581]
  - Cluster 3: [5.02326697, -1.53075614, 6.54878007]
  - Cluster 4: [16.02762225, 7.64966694, -4.699508]

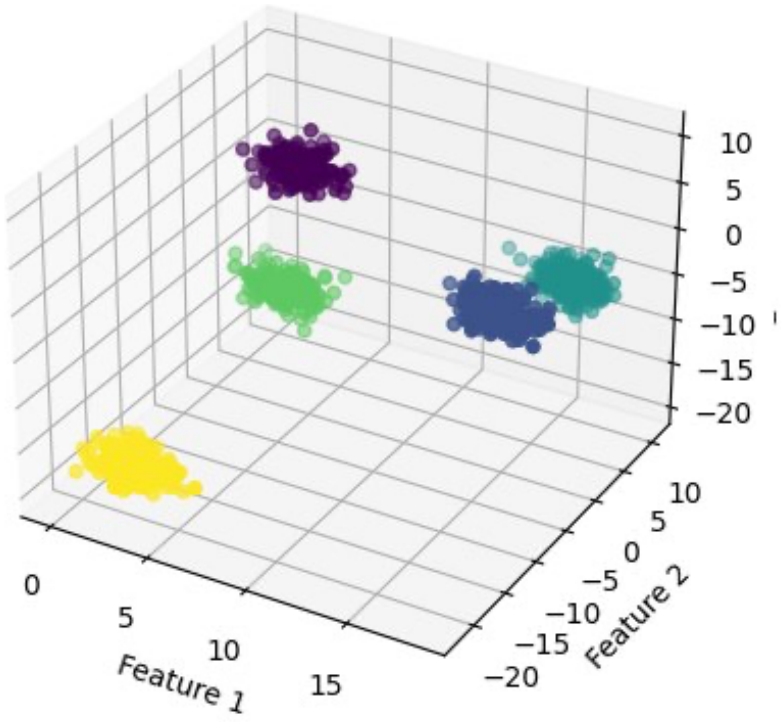

Figure 2: Dataset generated using scikit-learn library (1000 data points in 3 dimensions)

Comparing these results, we observe a strong similarity between the cluster centers generated by Hadoop and scikit-learn. Although there might be slight variations in the decimal values, the overall positions of the clusters are consistent. This similarity indicates that the Hadoop implementation successfully approximates the cluster centers generated by the scikit-learn implementation.

These findings provide evidence that the K-means algorithm implemented in Hadoop is effective in clustering. The Hadoop implementation demonstrates comparable results to a widely-used machine learning library like scikit-learn, which further validates its functionality and correctness.

The performance and accuracy of the Hadoop implementation is influenced by the initialization method, convergence criteria, and data preprocessing techniques used.

4. Summary

The implemented K-means clustering algorithm using the MapReduce framework proved to be an effective solution for clustering large datasets. The algorithm efficiently grouped data points into clusters based on their similarity, and the convergence behavior was achieved within a reasonable number of iterations.

The design and implementation of classes such as Point, Centroid, KMeansMapper, KMeansCombiner, and KMeansReducer allowed for modular and efficient execution of the algorithm. The experimental results confirmed the algorithm's scalability and its ability to handle big data applications.

Overall, the K-means clustering algorithm implementation serves as a valuable tool for various data analysis and clustering tasks, enabling efficient processing and clustering of large datasets.