The k-means Clustering Algorithm in MapReduce

The problem of partitioning a set of unlabeled points into clusters appears in a wide variety of applications, such as:

- *customer segmentation* – to target specific clusters/groups of customers;
- *document segmentation* – to cluster documents in categories based on tags and content;
- *image segmentation* – to categorize an image among different groups.

One of the most well-known and widely used clustering algorithms is Lloyd's algorithm, commonly referred to simply as the *k-means clustering* algorithm. The popularity of k-means is due in part to its simplicity – the only parameter which needs to be chosen is k, i.e., the desired number of clusters – and also its speed.

Let $X = \{x_1, \dots, x_n\}$ be a set of n data points, each with dimension d. The *k-means problem* seeks to find a set of k points (called *means*) $M = \{\mu_1, \dots, \mu_k\}$ which minimizes the function:

$$f(M) = \sum_{x \in X} \min_{\mu \in M} \|x - \mu\|_2^2$$

In other words, we wish to choose k means so as to minimize the sum of the squared Euclidean distances between each point in the data set and the mean closest to that point.

Finding an exact solution to this problem is NP-hard. However, there are a number of heuristic algorithms which yield good approximate solutions. The standard algorithm for solving the k-means problem uses an iterative process which guarantees a decrease in total error (value of the objective function $f(M)$) on each step. The algorithm is as follows:

Algorithm 1: The k-means clustering algorithm.

```
Input  : the set $X$ of $n$ data points in $d$ dimensions $X = \{x_1, \dots, x_n\}$
         the number $k$ of clusters to find
Output: the set $M$ of $k$ means in $d$ dimensions $M = \{\mu_1, \dots, \mu_k\}$
k-Means$(X, k)$ :
1     The means $\{\mu_1, \dots, \mu_k\}$ are randomly sampled from $X$
2     while a stopping criterion has not been met do
3         for each mean $\mu_i \in M$ do
4             $\omega_i \leftarrow \{\}$
5         for each data point $x_i \in X$ do
6             $c \leftarrow \mathrm{argmin}_j \|x_i - \mu_j\|_2^2$
7             $\omega_c \leftarrow \omega_c \cup \{x_i\}$
8         for each mean $\mu_i \in M$ do
9             $\mu_i \leftarrow \frac{1}{|\omega_i|} \sum_{x \in \omega_i} x$
10    return $\{\mu_1, \dots, \mu_k\}$
```

In other words, the k-means algorithm chooses k initial means $\mu_1, \ldots, \mu_k$ at random from the set X (line 1). Then, for each point $x \in X$, it finds the closest mean μ_c and adds x to a set ω_c (lines 5-7) initially empty (lines 3-4). Then, for each mean μ_i, it recomputes the mean value to be the centroid of the data points in ω_i (lines 8-9). These steps are repeated until the means have converged (line 2). The convergence criterion is typically when the total error stops changing between steps, in which case a local optimum of the objective function has been reached. However, some implementations terminate the search when the change in error between iterations drops below a certain threshold. Each iteration of this algorithm takes time $O(nkd)$. In principle, the number of iterations required for the algorithm to fully converge can be very large, but on real datasets the algorithm typically converges in at most a few dozen iterations.

In this project you must:

1. design a MapReduce algorithm (using pseudocode) to implement the k-means algorithm;
2. implement the designed MapReduce algorithm using the Hadoop framework;
3. test the implementation on a synthetic or real-world dataset;
4. write a short project report detailing your design, implementation and experimental results.

For higher marks, please address efficiency issues in your implementation; examples include, but are not limited to, the following:

- use combiners and/or more than 1 reducer;
- use custom `WritableComparable` objects;
- use the `Mapper` and `Reducer` classes `setup` and/or `cleanup` methods;
- test your implementation on a range of different datasets, by varying values of n, d, and k.[1]

[1]For example, you can write a small program taking as input n, d and k to generate different random sample datasets.