Big Data Analytics (SOEN 498/691)
Laboratory sessions

Tristan Glatard
Department of Computer Science and Software Engineering
Concordia University, Montreal
tristan.glatard@concordia.ca

February 7, 2017

Contents

Part I
k-means

1 Introduction

Clustering is a technique used in a variety of applications to categorize elements of a data set. In this lab, we will implement the k-means clustering algorithm in MapReduce. k-means is a very popular clustering algorithm due to its simplicity. You don't have to submit anything after this lab, it is only here to help you.

Given a dataset consisting of n elements and an integer $k \leq n$, the k-means algorithm works as follows:

1. Initialization: select k initial means m_1, ... , m_k. Means are also called centroids.
2. Assignment: assign every element x of the dataset to a set S_i such that $i = \text{argmin}_{i \in [\![1,k]\!]}(d(x, m_i))$, where $d(.,.)$ is a distance function defined on the data set.
3. Update: update m_i as the mean of the elements in S_i.

Repeat steps 2 and 3 until the m_i no longer change.

FIgure 1 illustrates the input and output of the k-means algorithm on a dataset of 1,000 points clustered in 4 clusters.

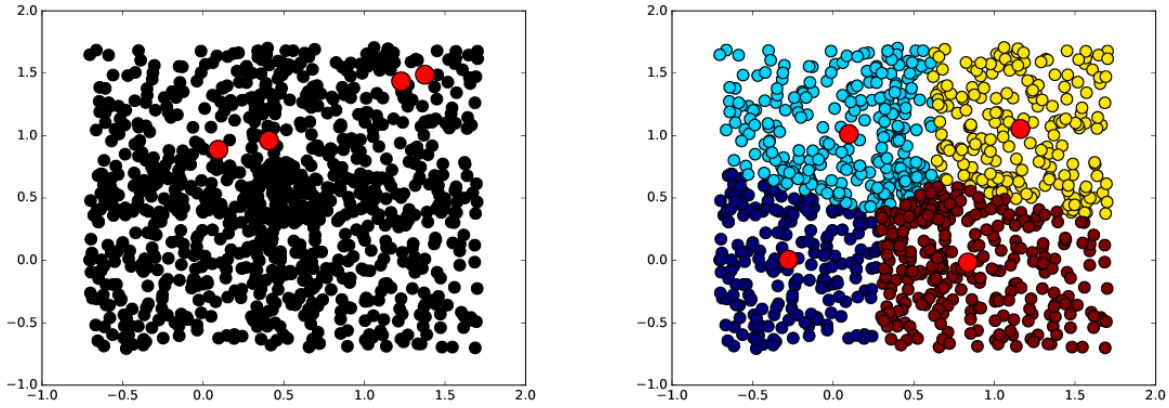

Figure 1: Illustration of k-means algorithm. Left: input dataset (black) and centroid initialization (red). Right: classification (the 4 different clusters are represented in cyan, blue, yellow and brown) and final centroid locations (red). The algorithm converged after 22 iterations.

2 Basic idea of the MapReduce implementation

We focus on a dataset containing 2D points. We will implement *each iteration* of the k-means algorithm as a MapReduce job. In other words, your program will run several MapReduce jobs, one for each iteration of the k-means algorithm. The loop and convergence detection will be handled in the main method (not in a MapReduce job).

Here is a description of the Map and Reduce functions:

- Map step:
  - Receive (_,v): _, (x,y)
    where x and y are the coordinates of a point in the dataset.
  - Emit (k,v): i, (x,y)
    where i is the index of the closest centroid to (x,y).
- Reduce step:
  - Receive (k, [v]): i, [(x_1,y_1), (x_2,y_2), ...]
  - Emit (k,v): i, ($\bar{x}$, $\bar{y}$)
    where $\bar{x}$ and $\bar{y}$ are the means of the x_i and y_i, respectively. ($\bar{x}$, $\bar{y}$) will be the coordinates of centroid i.

The centroids will be written in a file distributed to the mappers through the distributed cache. Convergence of the algorithm will be detected when the hash of this file no longer changes. Every iteration will produce a list of k centroids. Once the algorithm converged, a final map job will be run to output the classification.

3 Implementation

Implement the k-means algorithm as described in the previous Section (or your own version!). Make sure that it produces a correct result on the example above (input data file is available here). Note that the result of the algorithm is dependent on the initialization. You may generate other datasets using a script such as generate.py and plot them using plot-result.py.

Here is a possible solution:

(Link to file)

Figure 2 shows result samples on various datasets.

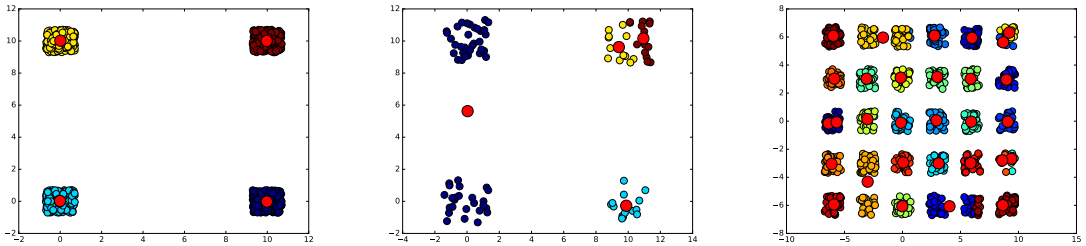

Figure 2: k-means results on various datasets clustered with k=4, k=4 and k=30 (left to right). Final centroids are in red and other colors represent clusters.

4 Bonus

Identify clustering errors that appear on Figure 2. Where do you think these errors are coming from? How can you improve this situation?