Project 2 of CSE 473/573

Shaoming Xu   UB# 50247057

Map Reduce K-mean on Hadoop

Q1. Algorithm description

The basic idea of Map Reduce framework is to use the (key, value) pair to divide a big job to many small jobs and delegate them to many machines. Each mapper or reducer will have a unique key. The value usually saves the data that we want to compute.

Map Reduce K-mean algorithm in this project includes four parts – driver, mapper, combiner, and reducer.

Driver is defined inside Kmean.java file. In driver, we set configuration, and generate and run job for each iteration of K-mean. We pass centers information to mapper in driver through conf.set() function. After each iteration, it will check if the K-mean is converged by checking the output from previous and current iterations. It will stop when converged.

The Mapper is defined inside KmeanMapper.java file. It includes two parts – setup() and map() functions. The setup function is used to read centers information from hdfs file system. The map function computes the Euclidean distances of the record to all centers. Then it sends the record to the reducer corresponding to the center with least distance.

The Combiner is defined in KmeanCombiner.java file. It lies between Mapper and Reducer in MapReduce framework. Generally, the Combiner will aggregate the output from mapper then send it to reducer.

The Reducer is defined in KmeanReducer.java file. The KmeanReducer will sum up the output from mapper, then calculate the mean as the reducer's output.

To make MapReduce work, we also need to make the class of value in (key, value) pair to be the subclass Writable Interface. In our work, we define RecordWritable class. It has pt and num fields. the pt is List<Double> type which saves the sum of records in axis=0. The num is int type, which determines the number of the records that sum up as pt. Typically, when num is 1, the object may be the output value of KmeanMapper or KmeanReducer. When num is greater than 1, the object must be the output of KmeanCombiner.

Q2. Result visualization

We follow these steps to do visualization.

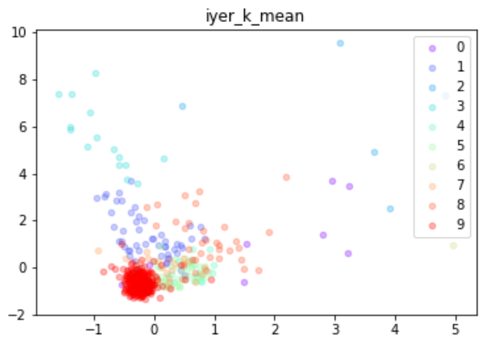
First, we generate K centers randomly from the data.

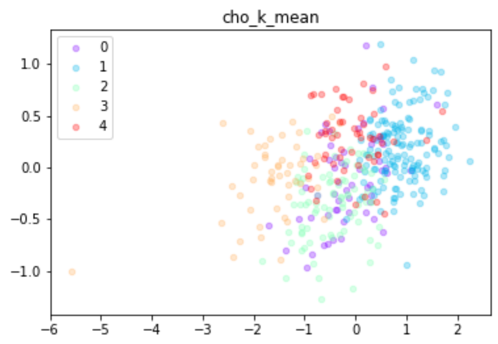
Second, we feed data and centers data to MapReduce Kmean. And get result centers.

Third we use the result centers to do clustering on data, get the line id of data.

Fourth, we do PCA on data.

Fifth, we use the data_PCA and index to visualize data.





Q3 Result evaluation, pro and con

We can see the images above, the MapReduce K-mean performs correctly for given data. The result vividly shows K-mean algorithm will generate sphere results.

K-mean has these strengths. First, efficient. The time complexity is O(n) because the numbers of iteration and centers usually are much less than the number of objects. Second, it is very easy to be implemented.

K-mean has these weaknesses. First, need to specify K. Without the knowledge of raw data, it is not easy to chose properly K. Second, initialization matters. Using different initial value of K centers will get different results. Third, empty clusters may occur. Luckily, we haven't meet this problem in our project.

MapReduce k-mean has these strengths. First, it can handle large data. Second, the hdfs file system helps to decrease the communication overhead in network. Third, we can use combiner to decrease communication over network further. Further the combiner can handle the performance issue caused on data distribution. It easy the work load for the reducer which center collect much more data than other reducers.

MapReduce k-mean has these weaknesses. First, a lot of read and write on file system. In each job, it need to read data for mapper, then to write data in reducer. To much read and write on disk may make it work slowly. I think it may be one of the reasons why people develop spark. Second, hard to make load balance. The difference of power on each machine and the strategy on key generating can affect it performance a lot.