Mining of Massive Dataset
Project 2 Report

Hao Jiteng, Zhou Lizhi, Yang Fangzhou

June 19, 2012

Abstract

K-means is a simple yet useful clustering algorithm. It's underlying natural implies that it could be parallelized or distributized. Some implementation of kmeans employs platforms such as Hadoop and CUDA to boost the process of mining of massive dataset. In our project we implement k-means algorithm on Apache Hadoop Project. We ran our algorithm on our tiny cluster. Evaluation has been done to measure our algorithm.

1 Introduction

1.1 Hadoop

Following is the introduction of Hadoop on its project homepage [**?**].

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-avaiability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

We mainly use the Hadoop MapReduce framework rather than HDFS.

1.2 k-means

There are many materials that introduce k-means algorithms. MacQueen, J. et al. proposed k-means algorithm in [**?**]. In [**?**] the author introduced a simple k-means MapReduce algorithm.

The dataset is relatively large comparing to the memory size. Thus we need a mechanism to deal with the incompatibility of memory. MapReduce

is a solution to this problem. The detail of our algorithm is documented in the following section.

1.3 Dataset

The MSRA image data set contains 295 classes of images. Every image class has hundreds of images. 7 features are used to describe the images. There are files that decribe the relavence between the image and the topic.

For our situation, we combine the 7 features into one and regards it as a vector with about a thousand dimensions. We did't use the relavence information for simplicity. The evaluation results are shown in Section. **??**

2 Clustering Algorithm

In this section we discuss the k-means clustering algorithm used in our implementation. Note that to find proper initial clusters, which may lead to fewer iterations and good result, we also implemented a Canopy clustering MapReduce algorithm.

2.1 Canopy Clustering

Canopy method is a relative easy but efficient clustering method. The basic idea of Canopy is to put the similar items into the subset, which are called Canopies. Different Canopies can overlap each other, but there is no node, which doesn't belongs to any of Canopy. Thus, here we use this method to pretreat the dataset before Kmeans method.

The algorithm is built up by the following MapReduce phases.

2.1.1 Procedure

1. Mapper
   `<LongWritable, Text>`$\rightarrow$ `<1,CanopyCluster>`
   Choose two distance threshold $t1$ and $t2$, where $t1 > t2$. For each node P in the list, calculate the distance between node P and Canopy, if this distance is less than t1, put P into this Canopy cluster, if no such kind Canopy cluster exits, take P as a new Canopy. If P was near enough to some Canopy cluster, which means the distance between P and the Canopy is less than t2, delete node P from the list.

2. Reduce
   `<1,CanopyCluster>` $\rightarrow$ `<identifier,KmeansCluster>`
   There is only one reducer and the reducer's job is to do another canopy custering on the output of the mappers described above. That's why the mappers always assign 1 to the keys of every output pair. For this round of clustering we can use another $t1, t2$ combination, namely $t3$

and $t4$. It's handy to use the KmeansCluster as the result format and kmeans clustering pass could load the result quickly.

2.2 k-means Clustering

Mahout Project [**?**] is a data mining framework under Apache Fundation. It contains a k-means implementation. The blog [**?**] gives a very detailed view of the algorithm. Our algorithm mainly based on the idea of Mahout Project. Actually, this algorithm is very similar to the BFR algorithm in our lectures.

The algorithm is built up by the following MapReduce phases,

1. The k-means iteration, output every cluster centroid if converged.
2. Assign every point to a known cluster and output the result.

The detail of the first phase is described as follows

2.2.1 Procedure

1. Mapper
   `<LongWritable, KmeansCluster>` → `<LongWritable,KmeansCluster>`.
   It reads the input file content as value. The key is the value offset in the file. Then it convert the value to a VectorDoubleWritable, which is used to represent the feature vector. It finds the nearest cluster to the vector, and output cluster id as key, a new cluster containing only point as value.

2. Combiner
   `<LongWritable, KmeansCluster>` → `<LongWritable, KmeansCluster>`.
   It reads the output from Mapper and combine those tuples who have the same cluster id(meaning that they are assigned to the same cluster) using KmeansCluster.omitCluster(). This function reduced the network transmission flow because the actual meaningful information need to be communicated between different nodes are only the N, SUM and SUMSQ of clusters, which is described in the lecture slides of BFR algorithm.

3. Reducer
   `<LongWritable, KmeansCluster>` → `<LongWritable, KmeansCluster>`.
   It reads the cluster id as key and the KmeansCluster as value. It adds the N, SUM and SUMSQ. The result leads to the combination of clusters. Finally the reducer outputs the result clusters of this single iteration. This clusters are input of next iteration. During reducer, if the movement of one cluster if less than a threshold, it's said to be "Converged". If a cluster is converged, a counter in context will increase by one.

4. If, in the driver, the counter equals the number of clusters, meaning that all clusters are converged, this phase is finished.

2.2.2 Assign Point to Clusters

After the iterations, the clusters are stable. Next we are going to assign every point to its nearest cluster. This phase only require mapper to do all the works, since this procedure is highly parallelized. Each two point are independent of each other.

The mapper takes the following form:
`<LongWritable, Text>→<Text, Text>`
It takes the data offset in file as key and the data as value. First it convert the Text object into a VectorDoubleWritble object that represents a data point. For each point we find the nearest cluster and output the cluster id and some other useful information for statistic and evaluation as value.

3 Evaluation

To evaluate the results of our experiment, we calculate the precision and recall of the results, then calculate the F value. Better results we get will lead to a better score.

3.1 Evalutation Dataset

Since the whole MS image dataset is too large, we pick a subset of the files but each item contains all features. Here is the file list:

Also, we use a set of different convergence threshold to evaluate the results and the running time.

3.2 Cluster Information

To run our Hadoop, we established a tiny cluster with 3 computer connected by a 100Mbps TP-Link router using Ethernet.

1. Namenode/DataNode/TaskTracker
    - Intel Core 2 Duo P8700 @ 2.53GHz
    - 2 Core 2 Threads
    - 4G DDR3 1066 RAM
    - OpenSUSE 12.1 x64
2. DataNode/TaskTracker
    - Intel Core 2 Duo T6600 @ 2.2GHz

- 2 Core 2 Threads
- 2G DDR2 800 RAM
- Archlinux i386

3. DataNode/TaskTracker

   - AMD Processor @ 3.0GHz
   - 6 Core 6 Threads
   - 8G DDR3 1066 RAM
   - Ubuntu 12.04 LTS x64

3.3 Evaluation Results

Results are listed in the table.1

THRESHOLD	PRECISION	RECALL	F	ITERATION	TIME

Table 1: Evaluation Results

3.4 Note on the Results

The results of given dataset are really poor, but we guess the dataset can be a problem because the features are actually not relarant to the labels.

We have run some testings on some different datasets whose points can be distinguish clearly. Our program gave almost perfect results.