

Homework 4 – K-means Clustering of Netflix Data

CS15-505 – Fall 2007

Assigned: 10/18/2007

Due: 11:59 pm on Thursday, 11/8/2007

(Start working on this assignment early **)**

Reading

Canopy Clustering - www.kamalnigam.com/papers/canopy-kdd00.pdf

K-means Clustering - en.wikipedia.org/wiki/K-means_algorithm

Goal

Cluster the Netflix movies using K-means clustering. You're given a set of movies and a list of which review which rater has given to which movies. Your goal is to create four hundred or so sets of related movies. Getting there should impart on you a respectable degree of map-reduce functions.

Data Set

Netflix data can be found in /shared/netflixprize.

The netflixprize data files are organized as follows:

- One movie data per line,
- One line is formatted as <movie-id>:(<rating>:)* (line components are separated by colons), where <rating> ::= <rater-id>,<rating>,<date>

Overview

K-means clustering is one of the simplest clustering algorithms, called k-means because we iteratively improve our partition of the data into k sets. We choose k initial points and mark each as a center point for one of the k sets. Then for every item in the data set we mark which of the k sets it is closest too. We then find the average center of each set, by averaging the points which are closest to the set. With the new set of centers we repeat the algorithm. However, if for each iteration we compare each point to each possible center, then predictably, this doesn't scale very well so we will make use of canopy clustering to reduce the number of distance comparisons we need to make. As discussed in the reading we create a set of overlapping canopies wherein each data item is a member of at least one canopy. We then revise our original clustering algorithm – rather than comparing each data point to each k-set-center, we only compare it to the centers that are located in the same canopy. This vastly reduces the amount of computation involved, the caveat is that if two points never occur in the same canopy they will not occur in the same final cluster.

Applications

news.google.com, clusty.com, Amazon.com related products, etc

Suggested Steps (each is a MapReduce)

- Data prep. In a sense, you can treat the rater IDs in a document as the terms in the document. In order to help you compute the number of rater IDs in common for canopy selection, we have created an inverted index for you, using rater IDs as the “terms”. The inverted index can be found in /user/paichul/netflixprize/index.

The netflixprize inverted index data files are organized as follows:

- One rater ID data per line,
- One line is formatted as <rater-id> (<rating>)* (line components are separated by colons), where <rating> ::= <movie-id>:<rating>

•For example, in this line

“1000053 8643:5,14916:5,13506:5,16860:5,16593:4,14274:5,13468:5,12397:5”,
1000053 is the rater ID, and 8643 is the movie ID, and 5 is the assigned rating.

•Canopy selection.

1. Read canopy clustering paper and any relevant wikipedia articles. Use the algorithm from the paper.
2. Canopy selection requires a simple distance function; we used the number of rater IDs in common. It also uses close and far distance thresholds; we used 8 and 2.
3. You use a cheap distance metric for the canopy clustering and a more advanced metric for the K-means clustering. We used vector cosine distance for our more advanced distance metric.

•Mark data set by canopy

1. Create a second data set in which movie vectors are also marked by canopy.
2. The suggested initial data structure looks like this:
movieID: {rater17: 1 rater21: 3} x {canopies: 1, 2, 7}
centroid: {centroid 1: canopy 2 <rater1: 3.12, rater2: 4.67>}

movieID is the key, and its values are raterID and their ratings, and its associated canopies.
centorid is the key, and its values are its associated canopy, raterID, and their ratings.

• K-means iteration.

1. Use the canopy centers as the source of initial K-means.
2. For Map step, for each movie, use the data to find the centroids in its canopies. The output includes a key:value pair where the key is the centroid, and its values are the movieVector. For the k-means-iter map-reduce we found it easiest to map over the entire data set and load the current k-mean-set-centers into memory during the configure() call to the mapper.
3. For Reduce step, average all movies per centroid and output the recomputed centroid. In the k-means-inter reduce where the new centers are found by averaging the vectors we found it necessary to minimize the number of features per vector, allowing them to better fit into memory for the next map. To do this we picked the top hundred thousand occurring raterIDs per movie and only output the average of those ratings.
4. Iterate through steps 2-3 four or more times until the centroids do not change much. You can set a threshold to define this change. If the change is too small, then stop.

Notes

1. Understand the whole process before you start coding. Read the canopy clustering paper and any relevant wikipedia articles before asking questions.
2. You might want to use reporter.SetStatus() to output basic debug information, such as how many k-centers were loaded into memory, the number of points mapped to so far, or the number of features per vector.

What to Turn In

1. Describe your data structures used at each MapReduce step.

2. Describe your canopy selection algorithm. What problems did you encounter? What are its strengths and weaknesses compared to potential alternatives? How did you do it using MapReduce or Can it be done using MapReduce? Why or Why not?
3. Describe how you implement K-Means using MapReduce. What problems did you encounter? What are its strengths and weaknesses?
4. Compare the two distance metrics and argue why one is cheaper than the other.
5. Report the Wall Clock time each step takes:
 - a. Canopy Selection
 - b. K-Means
6. Your .java files with comments and documentation detailing how to run your code (arguments, expected input, etc).