Machine Learning using MapReduce

What is Machine Learning

- *Machine learning is a subfield of artificial intelligence concerned with techniques that allow computers to improve their outputs based on previous experiences (stored as data).*
- Closely related to data mining and often uses techniques from statistics, probability theory, pattern recognition, and a host of other areas.
- *"More data usually beats better algorithms."* Anand Rajaraman.
- Sample uses of machine learning:
  - Fraud detection
  - Stock market analysis
  - Game playing
  - Spam detection in email
  - Recommendation of new products based on past purchases (Amazon, Netflix etc)
  - Find all similar news articles on a given day
  - Automatically categorize web pages according to genre (sports, economy, war etc)

Types of Machine Learning

- *Supervised learning is tasked with learning a function from labeled training data in order to predict the value of any valid input.*
  - Many algorithms are used to create supervised learners, the most common being Neural Networks, Support Vector Machines, and Naive Bayes classifiers.
- *Unsupervised learning is tasked with making sense of data without any examples of what is correct or incorrect.*
  - It is most commonly used for clustering similar input into logical groups. It also can be used to reduce the number of dimensions in a data set in order to focus on only the most useful attributes, or to detect trends. Common approaches to unsupervised learning include K-Means, Hierarchical clustering, and Self-organizing maps.

Sample Machine Learning Tasks

- Collaborative filtering
- Clustering
- Categorization

Collaborative Filtering

- Collaborative filtering (CF) is a technique, popularized by Amazon and others, that uses user information such as ratings, clicks, and purchases to provide recommendations to other site users.
- Four ways of generating recommendations are typical:
  - User-based: Recommend items by finding similar users. This is often harder to scale because of the dynamic nature of users.
  - Item-based: Calculate similarity between items and make recommendations. Items usually don't change much, so this often can be computed offline.
  - Slope-One: A very fast and simple item-based recommendation approach applicable when users have given ratings (and not just boolean preferences).
  - Model-based: Provide recommendations based on developing a model of users and their ratings.
- All CF approaches end up calculating a notion of similarity between users and their rated items.

Clustering

- Given large data sets, whether they are text or numeric, it is often useful to group together, or cluster, similar items automatically.
- Like CF, clustering calculates the similarity between items in the collection, but its only job is to group together similar items. In many implementations of clustering, items in the collection are represented as vectors in an n-dimensional space.
- Given the vectors, one can calculate the distance between two items using measures such as the Manhattan Distance, Euclidean distance, or Cosine Similarity. Then, the actual clusters can be calculated by grouping together the items that are close in distance.
- Popular approaches to build clustering include *k-Means* and *hierarchical clustering*.

Categorization

- The goal of categorization (often also called classification) is to label unseen documents, thus grouping them together.
- Many classification approaches in machine learning calculate a variety of statistics that associate the features of a document with the specified label, thus creating a model that can be used later to classify unseen documents.
- Features for classification might include words, weights for those words (based on frequency, for instance), parts of speech, and so on. Of course, features really can be anything that helps associate a document with a label and can be incorporated into the algorithm.

Distance Metrics

Represent a document as a vector of features.

- Euclidean distance.
- Manhattan distance.
- Cosine similarity: The cosine of the angle between two vectors.
- Mahalanobis distance: More sophisticated metric that is normalized and takes correlation into account.

Clustering techniques

- Canopy Clustering: Using canopy clustering the data is first partitioned into overlapping canopies using a cheap distance metric. The data is then clustered using more traditional clustering algorithms.
- k-Means Clustering: Commonly implemented using Lloyd's heuristic, which begins by partitioning data into k sets using some defined method or even arbitrarily. The mean point or centroid of each set is then calculated and the algorithm is repeated by associating each data point to the nearest centroid then finding the new set center. This is done repeatedly until convergence, which is determined by observing that the centroids no longer change upon successive iterations.
- Greedy Agglomerative Clustering: Builds the desired clusters from single data objects. A Greedy approach is to merge the two clusters with the greatest similarity at each step. This process is repeated until either the desired number of clusters is achieved or until the resulting clusters all meet some predefined characteristic.
- Expectation-Maximization Clustering: An iterative technique for estimating the value of some unknown quantity, given the values of some correlated, known quantity.

The Netflix Problem

- Rating set. Over 100 million ratings by more than 480,000 users on over 17,000 movies.
- Probe set. Around 1.4 million movie and user ids for which predictions must be made. Actual ratings are provided, so we can calculate the RMSE (root mean square error) rate.
- Input Format: One text file containing data for a single movie. Each of these files contains the movie identification number of the movie as the first line. Every subsequent line contains comma separated values for a rater identification number, an integer rating greater than or equal to one and less than or equal to five given by that rater, and the date on which the movie was rated. For example, 11674 (The Name of a Rose):

```
11674:
1331154,4,2004-08-02
551423,5,2004-07-19
716091,4,2005-07-18
```

Overview of MapReduce Solution

- **Step 1**. Data preparation.
- **Step 2**. Canopy selection.
- **Step 3**. Mark by canopy.
- **Step 4**. Expensive clustering.
  - *k-Means*
  - *Greedy Agglomerative*
  - *Expectation-Maximization*
- **Step 5**. Inverse indexer.

Step 1: Data Preparation

- Convert data to one line per movie:

  movieID_D `rater_i:rating_i,rater_j:rating_j,rater_k:rating_k,...`
  movieID_E `rater_u:rating_u,rater_v:rating_v,rater_w:rating_w,...`
  `...`

- For example the data for the movie "The Name of the Rose" will be transformed into the format:
  11674 `1331154:4,551423:5,716091:4,1174530:3,...`

Step 2: Canopy Selection

- Distance metric: if a set of z number of people rate movie A and the same set of z number of people rate movie B, then movies A and B belong to the same canopy.
- Using this metric it is possible that canopies may overlap, or in other words a movie may belong to multiple canopies. So long as each movie belongs to at least one canopy the necessary condition of canopy clustering will be met.
- Hence, in order for this to be true the value z must not be too large as the canopies may be large and many data points may lie outside of canopies.
- If the value of z is too small then the number of canopies will be few and each canopy will have many data points. Hence, the eventual expensive data clustering may not be very good.
- Output is the canopy centers with their ratings data (same format as the input)

Effect of distance on number of canopies created

Distance	Canopy Centers
5	75
6	108
7	169
8	251
9	326
10	433
11	530
12	685

Step 2: Canopy Selection (contd)

- **Map Step**: Every mapper maintains a collection containing the canopy center candidates it has determined thus far. During every map the mapper determines if each successive movie is within the distance threshold of any already determined canopy center candidate. If the mapped movie is within the threshold then it is discarded, otherwise it is added to the collection of canopy center candidates.
- The intermediate output sent to the reducer has the movieID as the key and the list of raterID-rating pairs as the value.
- **Reduce Step**: The reducer repeats the same process as the mappers. It receives the candidate canopy center movieIDs but removes those that are within the same threshold. In other words it removes duplicate candidates for the same canopy center.
- In order for this to work correctly the number of reducers is set to one.

Step 3: Mark by Canopy

- Mark each movie from the full data set from Step 1 with the identification number of the canopies it belongs to. The two inputs used for this step are the output from Step 1 and the output from Step 2. The same distance metric from Step 2 is used in this step to determine if the movie belongs to a particular canopy or not.
- The output will have the following format:

  movie_A `rater_i:rating_i,rater_j:rating_j,... ;canopy_U,canopy_V,...`

Step 3: Mark by Canopy (contd.)

- **Map Step**: Each mapper will load the canopy centers generated by Step 2. As each movie is received from the full data set the mapper determines the list of canopy centers that the movie is within, using the same distance metric from Step 2. The intermediate output is the movieID as the key and its raterID-rating pairs and list of canopies as the value.
- **Reduce Step**: The reducers simply output the map output.

Step 4: Expensive Clustering: k-Means

The expensive clustering steps do not change the full movie data set. They merely move around the canopy centers so a more accurate clustering is determined.

- The K-means clustering step is performed iteratively until convergence is achieved. In simple terms this means until the k-centers no longer change. However, in practice this can take an incredible amount of time or never be achieved at all. So for testing purposes the algorithm was run iteratively up to five times and the final result considered converged.
- The expensive distance metric used in this step is cosine similarity.
- The two input data sets for this step are the full data sets marked with canopies created by Step 3 and initially the canopy centers created by Step 2. The output is a list with the new cluster centers (movieID) as the key and raterID-rating pairs list as its values in the same format as the output of the canopy selection MapReduce (Step 2).

Step 4: Expensive Clustering: k-Means (contd.)

- **Map Step**: Each mapper loads the k-centers from the previous K-means MapReduce into memory. For the first iteration the canopy centers generated by Step 2 are used. Each movie that is mapped is also contains a list of the canopies it belongs to. Using the expensive distance metric the mapper determines which canopy the movie is closest to and outputs the chosen canopyID as the key and the mapped movie as the value.
- **Reduce Step**: This step must determine the new center for every canopyID that it receives from the mapper. The process to do this involves determining the theoretical average movie in a canopy, then finding the actual movie that is most similar to this average value. When finding the average movie one determines the set of raters that belong to a canopy. For each of these raters it can be determined how they scored movies on average. With this information we can use cosine similarity to determine the movie most similar to this average movie.

Step 5: Inverse Indexer

- This phase outputs results that can be used. The aim is to associate each movie with a cluster center in an inverted index format. Hence, the cluster center movie identification numbers are used as the keys and the associated movie identification numbers are used as the values. The two inputs for this step are the list of centroids and the full movie data set. The output has the following format:

  movieID_centroid `movieID_A:similarityA,movieID_B:similarityB,...`

- **Map Step**: The map loads the cluster centers determined by any one of the algorithms from Step 4. For each mapped movie that is within the cheap distance metric from Step 2 of any cluster center, the similarity is calculated for that movie to the cluster center using the appropriate distance metric. The intermediate output sent to the reducer will have the cluster center as the key and the mapped movieID and similarity as the value.
- **Reduce Step**: The reduce simply concatenates the movieID-similarity pairs for each cluster center.

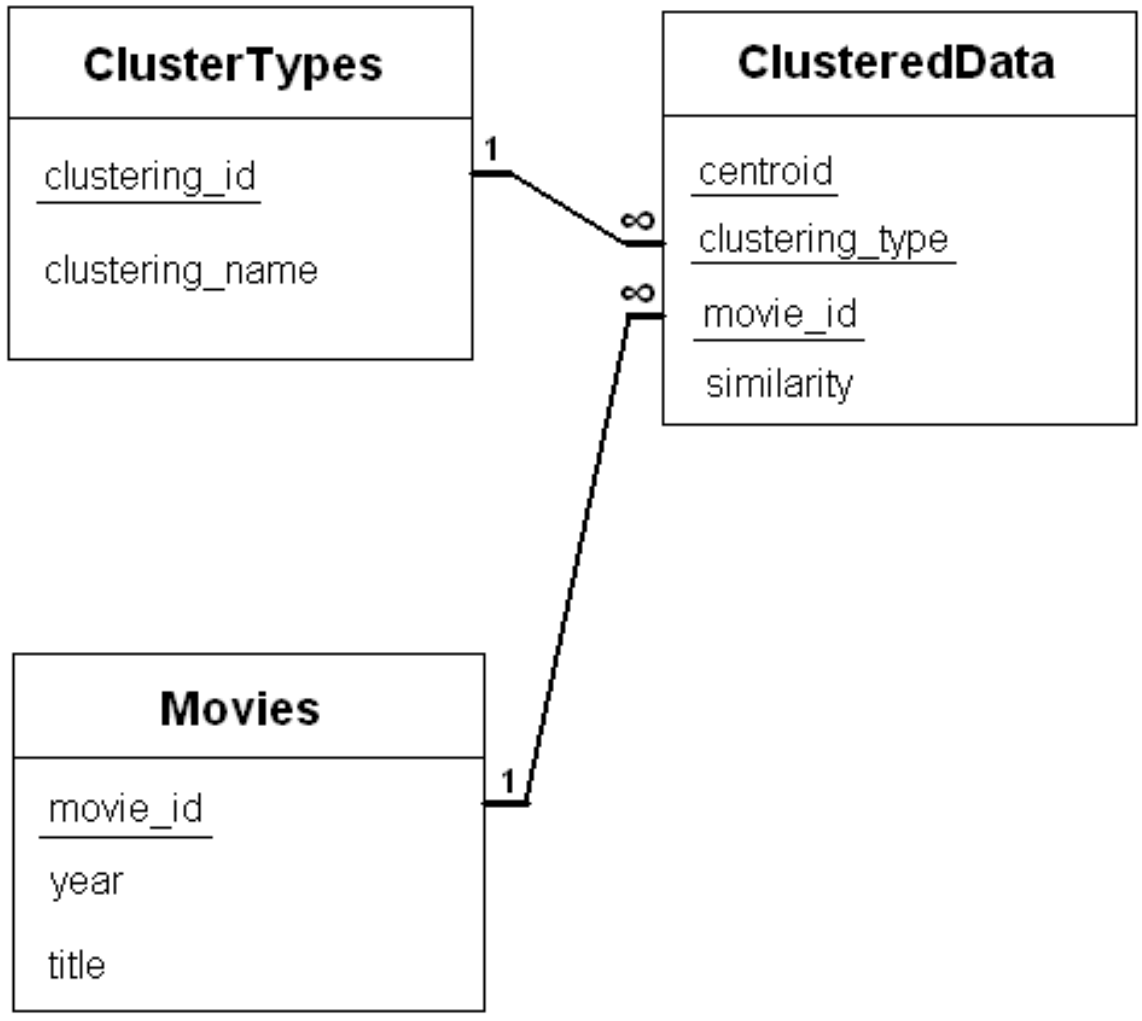
Predictor Data Preparation

To produce a list of a few similar movies for every movie and rater pair that we are interested in making a prediction for.

- The input to the map is the probe file. The output will have the format:

  probeMovieID `similarMovie1,similarMovie2,...`
- Uses a relational database in tandem to get information about the centroids. The database connection is opened during the initialize method for the map and closed during the cleanup method for the map.

Netflix database schema



Predictor Data Preparation (contd.)

- **Map Step**: The database is queried to find several similar movies using the following procedure:
  - If the movie is a cluster center then fetch several of the most similar to it.
  - If the movie is not a cluster center:
    - Find the cluster center it is closest to.
    - Fetch several movies most similar to this cluster center.

  The intermediate output sent to the reduce is keyed by movieID and the value is a list of the similar movies.
- **Reduce Step**: The reduce simply concatenates the similar movies for a particular probe movie.

Selected RMSE comparisons versus Cinematch.

Method	RMSE	vs. Cinematch
Random	1.9487	-104%
Expectation-Maximization	1.0808	-13.5%
Simple Mean	1.0540	-10.7%
Greedy Agglomerative	1.0378	-8.96%
K-means	1.0269	-7.81%
Cinematch	0.9525	0.00%
Pragmatic Theory	0.8596	+9.02%
Goal	**0.8572**	**+10.0%**

Cinematch is Netflix's own algorithm and *Pragmatic Theory* was the algorithm that won the contest.

Apache Mahout

Apache Mahout is an open-source machine learning library that is mostly implemented on top of Hadoop. Currently it is based on four use cases:

- *Recommendation* mining takes users' behavior and from that tries to find items users might like.
- *Clustering* takes e.g. text documents and groups them into groups of topically related documents.
- *Classification* learns from existing categorized documents what documents of a specific category look like and is able to assign unlabelled documents to the (hopefully) correct category.
- *Frequent itemset mining* takes a set of item groups (terms in a query session, shopping cart content) and identifies which individual items usually appear together.

Clustering With Mahout

- Prepare the input. If clustering text, you need to convert the text to a numeric representation.
- Run the clustering algorithm of choice using one of the many Hadoop-ready driver programs available in Mahout.
- Evaluate the results.
- Iterate if necessary.

Data is often represented as a vector, sometimes called a feature vector. In clustering, a vector is an array of weights that represent the data.
Mahout comes with two Vector representations: `DenseVector` and `SparseVector`

References

- *Data Clustering Using MapReduce*, Masters in Computer Science project, Makho Ngazimbi, 2009. http://cs.boisestate.edu/~amit/research/makho_ngazimbi_project.pdf
- *Introducing Apache Mahout*, Grant Ingersoll, Chief Scientist, Lucid Imagination. http://www.ibm.com/developerworks/java/library/j-mahout/
- *Apache Mahout*. http://mahout.apache.org/
- *Map-Reduce for Machine Learning on Multicore*. http://www.cs.stanford.edu/people/ang/papers/nips06-mapreducemulticore.pdf