Project Progress Report-2

Empirical Analysis of Predictive Algorithms for Collaborative Filtering

Dhayalini Nagaraj

A20359686

Abstract:

Collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users. In this project, we use different algorithms for this task based on correlation coefficients, similarity calculations to predict the users preferences. We analyse and compare different algorithms based on accuracy in a set of problem domains.

Experiments:

Recommender systems are build using the recommendation inputs given by the people. Then the system aggregates and directs to appropriate recipients. It produces individualized recommendations and has the effect of guiding the user in a personalized way.

Recommender systems typically produce a list of recommendations in one of two ways through collaborative and content-based filtering or the personality-based approach.

- Collaborative filtering approaches building a model from a user's past behaviour as well as similar decisions made by other users. This model is then used to predict items that the user may have an interest in.
- Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties.

In this paper, we focus on collaborative filtering techniques. The Books product and the Ratings dataset which we use for this project includes no metadata or reviews. It includes only User, item, rating and timestamp tuples. Recommendation system can be built using collaborative filtering approach for this particular dataset. A variety of algorithms have been reported and evaluated empirically. Some of the most popular algorithms in collaborative filtering use a correlation-based approach. We will be evaluating three types of algorithm in this project and they are **Memory-based algorithms, Extensions to memory based algorithms and Model based algorithms**. Memory-based algorithms operate over the entire user database to make predictions. In Model-based collaborative filtering uses the user database to estimate or learn a model, which is then used for predictions.

Approaches considered for our project and its implementations:

Collaborative filtering uses implicit or explicit user votes to build the system**.** Explicit voting refers to a user consciously expressing his or her preference for a title, usually on a discrete numerical scale. In book dataset, the ratings scale is from 1 to 5, where the user rate each book explicitly based on

the scale. Implicit votes are based on other browsing data, purchase history and other types of information. Since we are using explicit votes for our project, the amount of missing vote will be nil and the dataset is also huge.

In memory based algorithms in order to measure similarity correlation between two user is found. This gives us a value from -1 to 1 which determines who alike two users are. A value of 1 means that they both rate in the exactly the same manner, whereas a value of -1 means that they rate things exactly opposite. We use Pearson correlation coefficient. The basic correlation algorithm used to measure the ratings and compare them. They will get a positive correlation if they vary in the same way else negative correlation. The second similarity measurement is called vector similarity. Treat two users as vectors in n-dimensional space where n is the number of items in the dataset. Using the two vectors, compare the angle between them. If it points to the same direction it gives positive similarity else negative similarity. The extensions to memory based is done using default voting and inverse user frequency and case amplification. Default voting simply adds a number of imaginary items that both have rated in common in order to smooth the votes. Inverse-user frequency, we just transform each vote when weighting two users such that commonly rated items are given less importance.

Model based algorithm is building a model from the dataset of book ratings and use that model to build the recommendation system without using the complete dataset. We use Probability of predicting method. Bayesian networks and clustering is used for this algorithm.

Implementation and Results:

The part that I am working on this project is Model based algorithm. The votes are integer valued with a range for 0 to m we have

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \Pr\left(v_{a,j} = i | v_{a,k}, k \in I_a\right)\ i$$

The probability expression is the probability that the active user will have a particular vote value for item j given the previously observed votes.

$$\Pr\left(C = c, v_1, \ldots, v_n\right) = \Pr(C = c) \prod_{i=1}^{n} \Pr\left(v_i | C = c\right)$$

Certain users have a common set of preferences and tastes. From the class, the preference for each item differs. The probability model is then built using the probability of class and the votes.The parameters of the model, the probabilities of class membership Pr(C = c), and the conditional probabilities of votes given class Pr (vi |C = c) are estimated from a training set of user votes, the user database.

The similarities between the data sets has to be measured in order to build the model based on the probabilities. The similarities between the items can be found using the below code.

#Dictionary of items that are similar to each other,

```
def calculateSimilarItems(prefs,n=10):
result = {}
 #Invert the preference matrix to be item-centric
 itemPrefs = transformPrefs(prefs)
 c=0
 for item in itemPrefs:
   #Status updates for large datasets
   c+=1
   if c%100==0:
     print "%d / %d" % (c, len(itemPrefs))
   #Find the most similar items to this one
   scores =
   topMatches(itemPrefs,item,n=n,similarity=sim_distance)
   result[item] = scores
 return result
```

Iterate through the items and calculate the similarities between the items. The data set of similarities between items can be now used to build the recommender system without using the whole dataset. The below code can be used for this.

```
def getRecommendedItems(prefs, itemMatch, user):
 userRatings = prefs[user]
 scores = {}
 totalSim = {}

 #loop over items rated by this user
  for (item, rating) in userRatings.items():

    #Loop over items similar to this one
    for (similarity, item2) in itemMatch[item]:

        #Ignore if this user has already rated this item
        if item2 in userRatings:
        continue
        #Weighted sum of rating times similarity
        scores.setdefault(item2,0)
        scores[item2] += similarity * rating
        #Sum of all the similarities
        totalSim.setdefault(item2,0)
        totalSim[item2]+=similarity
```

The experiments that has to be implemented in to build a cluster modal from the similarities dataset.

Evaluation:

Evaluation can be done by dividing the dataset into training and test dataset. After the filtering is done, can evaluate the accuracy of the system using absolute error and k-fold cross validations. We are not using the whole dataset to predict so the quality of prediction will vary depending on the cluster model that is build for the system.

I have to build a cluster modal from the similarities of the items and then evaluate the results in the following weeks.

Reference:

http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html
https://ashokharnal.wordpress.com/2014/12/18/worked-out-example-item-based-collaborative-filtering-for-recommenmder-engine/
http://aimotion.blogspot.com/2009/11/collaborative-filtering-implementation_13.html
https://www.codementor.io/jadianes/build-data-products-django-machine-learning-clustering-user-preferences-du107s5mk