Implementing Recommender Systems

Namrata Bajpai
Computer Science Department
Binghamton University
Email: nbajpai1@binghamton.edu

Abstract: This paper gives a brief introduction to the recommender systems and a design to implement the system. The recommender systems is a type of the information filtering system which provides a platform that seeks to predict the rating or a preference that a user may wish to give to an item. Examples include recommending the books, movies, CDs, products. This paper implements the design of the recommender system which makes use of the collaborative filtering with the Pearson coefficient as the similarity measure with the Top 50 users to make the missing rating predictions.

Introduction

Recommender systems are the software based tools and techniques which provide suggestions that are helpful to the users corresponding to their perspectives. These suggestions correspond to the different kinds of the decision making processes such as what items do we need to buy, what movies to watch, what music to listen to or what online items specific to a domain should we read.

Formal Model:

Let **X** be a set of customers and **S** be a set of the items. The utility function u can be defined as, u:
X x S->R
where R is set of ratings and is a totally ordered set. Examples include 0-5 stars, real number in [0,1].

Approaches used for the recommender system:

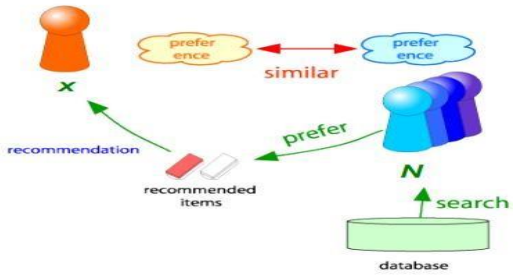
For the problems of gathering known ratings, extrapolating unknown ratings from the known ones, evaluating extrapolation method to measure success or performance of the system and the ***cold start problem*** where most items are not rated or they have no history below 3 approaches can be used.

Content Based: recommend items to customer x similar to the previous items rated highly by x. Example include recommending movies with the same actor. But the appropriate features are hard to find. Also, it never recommends items outside user's content profile.

Latent factor based: users and items are characterized by latent factors, each user and item is mapped onto a latent feature space. Each rating is approximated by the dot product of the user feature vector and the item feature vector. The prediction of the unknown ratings also uses this dot product.

Collaborative Filtering

Involves finding other users whose past rating behavior is similar to that of the current user and use their ratings on other items to predict what the current user will like. Besides the rating matrix R, a user–user CF system requires a similarity function $s: U \times U \rightarrow R$ computing the similarity between two users and a method for using similarities and ratings to generate predictions. This method works as follows,
Consider user x. Find the set N of other users whose ratings are similar to that of x. Estimate x's unknown ratings based on the ratings of the users in N. The model works as below,



The similarity measure used in this design implementation is Pearson correlation coefficient.

Sxy = items rated by both the users x and y.

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

$\overline{r_x}, \overline{r_y}$ ... avg. rating of x, y

There are several other methods like the ***Jaccard similarity measure*** which has the problem that it ignores the value of the rating. The ***cosine similarity measure*** calculated as, has the problem of treating missing ratings as negatives.The best and close approximates are provided by the Pearson's coefficient.

Design Logic:

This program uses user to user collaborative filtering with the use of Pearson coefficient as the similarity measure to predict missing ratings for the items.

Initially we have a matrix of dimensions ***"users(N)*no_of_items(M)"*** initialized to 0. Once the input file is provided by the user we fill the matrix from the input file and if the user has given no rating for a given item id its marked as 0 within the matrix.

The program takes the input file path, the number of the users and the number of items as the input from the command line.

We then calculate a mean rating mean_i for each user say i. This mean will be used to set the rating for an item in case of the cold start.

We are predicting the missing values based on the similarity calculated using the Pearson's coefficient using the formula previously described in the document. To achieve this we define a method "pearson" with below steps,

pearson(user1,user2)

1. *User 1 and user2 arr the arrays containing rating values by user1 and user2.*
2. *This method returns the pearson coefficient value between the two users.*
3. *Calculate the mean for user1 and user2 for common rated items as mean_u1 and mean_u2.*
4. *If the users have rated no common items then simply return 0.*
5. *Or else calculate the similarity based on the pearson coefficient formula documented.*

Sort the similarity map.

Further using the ***weighted mean*** for prediction ratings where weights being the similarity measures we fill out the missing ratings of an item for a user. Write the ratings predicted in the output file.

References:

https://web.stanford.edu/class/cs246/slides/07-recsys1.pdf
https://en.wikipedia.org/wiki/Recommender_system
http://www.hindawi.com/journals/aai/2009/421425/
http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf