Hybrid Recommendation System

Khagesh Patel
Dept of Mathematics
IIT Kanpur

Ankush Sachdeva
Dept of Computer Science
IIT Kanpur

Advisor: Dr. Amitabha Mukerjee
Dept of Computer Science
IIT Kanpur

April 24, 2014

Abstract

Recommendation systems are widely used in e-commerce companies like Amazon and Netflix to help users discover items that might interest them. Due to their wide applicability, recommendation systems have become an area of active research. We do a comparative study on the different algorithms used to do recommendation popularly and build a hybrid model out of them.

Introduction

Various algorithms for recommending choices to users have been used by companies involved in selling products and services online like E-commerce sites, movie renting sites, housing and restaurant recommendation sites etc. These algorithms can be majorly classified into two types content-based techniques and collaborative-filtering based techniques. The former rely on the computing the similarity of items based on the meta data provided about the items i.e. incase of movies data like the genre, date of release, IMDB rating etc. For example if the user has watched a lot of comic thrillers, he is recommended a suitable comic thriller. There are also demographic content based techniques which compute similar users based on their same demography to do recommendation. The collaborative filtering techniques rely on computing the similarity based upon similarity of the user ratings for multiple items i.e. if two users have liked similar movies in the past they are likely to have same preferences.
In collaborative filtering, we will also look at the latest techniques which incorporate the factor of time while recommending products. This is very essential especially, in todays time when the users have way too many choices and their preferences change frequently over time.

Dataset Used

We will be using the MovieLens 100k dataset for testing and training purposes. The reason for using this particular dataset is that the data is made available by a reputed and trustworthy source (University of Minnesota) and contains the information about the movies and the demographic data for users which we require in our content-based analysis. Also, each of the user-item ratings has a timestamp as the ratings have been collected over a period of 7 months.
The dataset contains 100,000 ratings from 943 MovieLens users on 1682 movies from September 19^{th}, 1997 through April 22^{nd}, 1998. We will use 80k for training and 20k for validation wherever such a partition is required.

Data Analysis

We analyzed the given to dataset to get some insight. Here are some of the results. The pie chart in Figure shows the overall distribution of ratings. The percentage of people voting for 3,4,5 is unexpectedly higher that people voting for 1 and 2. This shows that people have tendency to not rate a movie rather than rating it low if they do not like it.

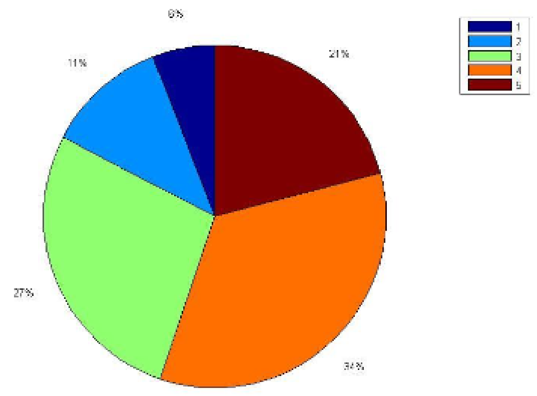

Figure 1: Distribution of ratings

Scatter plot in Figure show mean rating v/s variance scatter plot. We can see from the plot that very high rating(around rating 5) or very low rating(around rating 1) tend to have low variance compared to rating in middle range.So we can take end rating cases as consistent among different users.

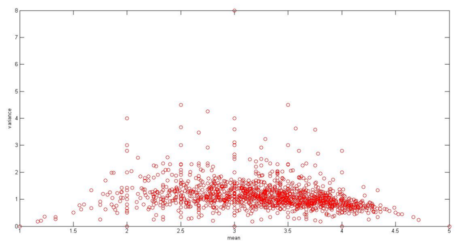

Figure 2: Mean v/s variance of rating

Since many entries in rating matrix are unknown we first filled these entries using SVD. Plots in Figure and are isomap and local linear embedding of rating matrix respectively. It is evident from these plots that rating from different users follow lie in a manifold.

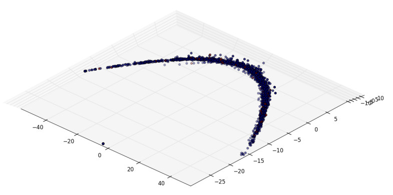

Figure 3: Isomap of rating matrix

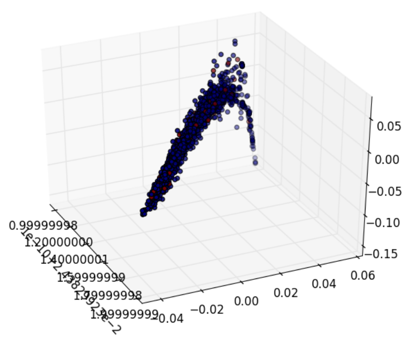
Figure 4: Locally linear embedding of rating matrix

Methods

Slope One

It is one of the simplest non trivial algorithm used for recommendation . It is based on the intuitive principle of popularity differential. To measure this difference, we take a pair of items which are rated by the same person and take the difference in rating.

For Slope One method we first calculate deviation matrix(D) given by

$$D_{j,i} = \sum_{u \in S_{j,i}} \frac{u_j - u_i}{\text{card}(S_{j,i})}$$

where u,u_j,$S_{j,i}$ denotes user, rating to item j by user u,set of all the users who rated both item j and i respectively. Then we can predict rating by:

$$r(u_j) = \frac{1}{\text{card}(R_j)} \sum_{i \in R_j} D_{j,i} + u_i$$

where R_j is set of all items such that user u has rated that item and item pair i and j is rated by atleast one more user.

KNN

We can apply this algorithm to either user feature vector or item feature vector giving rise to two separate algorithms: UserKnn and ItemKnn [1, 2]. This algorithm finds k nearest neighbour of given feature vector based on some similarity criteria and then take weighted average of this k neighbours. In common scenario more dense data is available for items therefore ItemKnn is generally a preferred choice.

We have used Pearson correlation coefficient as our similarity measure which for two vector X and Y can be given by:

$$\frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

Content based

In many recommender system you will come across user who have not rated significant amount of items. For these users methods like KNN and SVD who rely heavily on data perform poorly. This method suggest items to users based on content information of items for example in our case we can suggest movies based on genre similar to genre that a particular user generally prefers.

Demographic based

This method recommends items to the users based on their personal information like gender, age, profession and location. This method does not rely on rating system therefore this method generally tends to perform better then other method for new users with no rating.

Baseline Predictor

For the dimensionality reductions methods we first build a baseline predictor. We estimate a baseline prediction for the unknown rating of user u and item i.

$$\begin{aligned} & b_{ui} = b_i + b_u + \mu \\ \text{where} & \\ & b_i \text{ is the item bias} \\ & b_u \text{ is the user bias} \\ & \mu \text{ is the average rating of dataset} \end{aligned} \tag{1}$$

b_i and b_u calculate the deviation of item i and user u from the average item and user ratings respectively in the complete dataset.
For example, if the average rating of all movies if 3.3 and Godfather being an above average movie receives ratings 0.5 above the average. Also if user X is a critical user who tends to rates movies 0.2 below the average. The baseline prediction for X-Godfather pair would be $3.3 + 0.5 - 0.2$.

Singular Value Decomposition

In this method [2], we try to reduce the dimensionality of the given dataset by using Singular Value Decomposition. The matrices obtained after SVD are helpful in our case as it generates the best approximation of the input matrix in terms of Frobenius norm which minimizes the RMSE.
We build a user feature feature matrix and an item feature matrix of lesser dimensionality.

$$\begin{aligned} & \hat{r}_{ui} = p_u^T q_i + b_{ui} \\ \text{where} & \\ & \hat{r}_{ui} \text{is the predicted rating} \\ & p_u \text{ is user matrix} \\ & q_i \text{ is the item matrix} \\ & b_{ui} \text{ is the baseline prediction} \end{aligned} \tag{2}$$

Since, our input matrix is a sparse one we perform a stochastic gradient descent to generate our output matrices iterating over the given ratings and reducing the RMSE with each iteration.

$$\begin{aligned} min\Bigg(\sum_{(u,i)\in\kappa} \Big(r_{ui} - \mu - b_u - b_i - p_u^T q_i \Big)^2 + & \\ K\Big(b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2 \Big) \Bigg) & \\ \text{where} & \\ & \kappa \text{ is set of all given ratings} \\ & \text{K is the regularization constant} \end{aligned} \tag{3}$$

On MovieLens data we take learning rate to be 0.01, regularization constant to be 0.015 and maximum 400 iterations. We decompose the input matrix to rank 15 matrices.

α	0.001	0.01	0.03	0.05	0.1
RMSE	0.93412	0.92976	0.93053	0.93179	0.93354

Table 1: RMSE values for different values of $alpha$ in timeSVD++.

Incremental SVD

This is an improved version of the SVD [3]. Here we aim to integrate both explicit and implicit feedback of the user to do predictions. For most products implicit feedback could include data like purchase history, number of times the product link was clicked etc. However, in our dataset such information is not available. None the less, a different kind kind of implicit data is available to us. The dataset provides us the information on what movies were rated and what remained unrated by the user i.e. a binary matrix. This information does not look much useful at first however it has proved to considerably improve our recommendations.
We incorporate this implicit feedback into our user feature vector

$$\hat{r}_{ui} = q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{y \in N(u)} y_j \right) + b_{ui} \tag{4}$$

where

$N(u)$ is the set of items rated by user u

Here, every user is represented by a feature vector p_u as trained using explicit feedback as done in (2) and the sum $\sum_{y\in N(u)} y_j$ which incorporates the implicit feedback.We train this model using SGD with learning rate 0.004, regularization constant 0.005 and maximum 600 iteration. We decompose the input matrix to rank 5 matrices.

$$\begin{aligned} min\Bigg(\sum_{(u,i)\in\kappa} \Bigg(r_{ui} - \mu - b_u - b_i - \Bigg(p_u + |N(u)|^{-\frac{1}{2}} \sum_{y\in N(u)} y_j \Bigg) \Bigg)^2 + \\ K\Big(b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2 \Big) \Bigg) \end{aligned} \tag{5}$$

Incremental SVD with Temporal Dynamics

This technique is similar to SVD++ with the exception that it also involves the factor of time [3, 4]. Users change change their preferences over time. We divide the whole time space into a number of equally spaced time bins. Each ratings belongs to one time bin. We use a linear model to capture the preference drift in users. So if user u rated a movie in time bin t, the deviation is given by

$$\begin{gathered} dev_u(t) = sign(t - t_u).|(t - t_u)|^\beta \\ \text{where} t_u \text{is the mean date of rating by user u} \end{gathered} \tag{6}$$

Therefore, if we have f factors,

$$\begin{aligned} p_u^T &= (p_{u1}(t), p_{u2}(t), \ldots, p_{uf}(t)) \\ p_{uk}(t) &= p_{uk} + \alpha_{uk}.dev_u(t) \end{aligned} \tag{7}$$

α which determines the effect of the time-variant part of SVD is found to give optimal results at 0.01. Table 1 gives the RMSE value for different values of α. The number of time bins selected is dependent on how much data we have. If we take a finer resolution and we do not have enough data, our model will give a higher RMSE value. We have used 25 bins for our dataset which gave the most optimal model. Table 2 shows the RMSE values for different timebins.

TimeBins	5	10	25	40	60
RMSE	0.93220	0.93045	0.92976	0.93042	0.93103

Table 2: RMSE values for different values of time bins in timeSVD++. Larger the number of time bins, finer the resolution

The final model looks much similar to SVD++.

$$\hat{r}_{ui} = q_i^T \left(p_u(t) + |N(u)|^{-\frac{1}{2}} \sum_{y \in N(u)} y_j \right) + b_{ui} \tag{8}$$

where

$N(u)$ is the set of items rated by user u

We use the learning rate of 0.005 and increase the number of iterations to 1200 .We decompose the input matrix to rank 5 matrices.

Hybrid Model

We tried to build a simple model to verify our claim that a hybrid model could perform better than individual methods. We took a linear combination of these techniques - content-based collaborative filtering, Demographic based collaborative filtering, User-user k nearest neighbour and SVD with temporal effects. We achieved an improvement of 1% using this model. Weights that we used are 0.26,0.05,0.09 and 0.60 for User-user knn, content based method, demographic based method and SVD with temporal effect respectively.

Results

RMSE values for different algorithms. Lower the RMSE better the accuracy of the model.

Method	**RMSE**
Slope One	1.03136
KNN (user-user)	0.9439889
KNN(item-item)	0.9500658
Content Based	1.8461
Demographic Based	1.11833
SVD	0.942863
SVD++	0.93624
timeSVD++	0.929762
Hybrid	0.915856

Conclusion

We studied the various algorithms individually and observed that each algorithm has its own strengths. For users with sparse rating matrix, content and demographic based methods are suitable whereas for other cases KNN and SVD methods are more optimal. Incorporating the time factor in time based SVD had its own challenges. Using a simple model which just ignored the previous ratings of the user would have lost too much signal, therefore we used a linearly degrading model which gave better results. The given dataset had ratings spread over 7 months. In a bigger database like that of Netflix, the effect of time based model would have been more prominent.
The use of implicit feedback as simple as taking the boolean vector of whether a user has rated a movie or not had a considerable effect on the results. This is promising of even more exciting results when more concrete implicit feedback models like purchase history of the customer etc. are available.

Finally, combining KNN, Demographic, content-based and time-SVD++ methods using weighted mean, we achieved an RMSE of 0.915856 i.e. a 1% improvement over the best individual method. This change although appears to be small in magnitude, but has a large impact on the top 10 recommendations as emphasized by Koren himself [5].

References

[1] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[2] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[3] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at AT&T Research*, 2008.

[4] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[5] Y. Koren. Netflix forum - how useful is lower rmse.