Recommendation System for MovieLens

Team Member: Zining Wang, Jiayu Yao, Qiyang Li

Abstract

We used different models to predict movie ratings by different users. After comparing with variant models of K Nearest Neighbor (KNN), Singular Vector Decomposition (SVD) and XGBoost, we found out the modified SVD, which combines global and regional effects together, turned out to be the most accurate and fastest model, with an average testing RMSE of 0.8875.

I. INTRODUCTION

Recommendation systems suggest items of interest based on the user's preference. These systems have raised great interest in business fields, and they are widely used in lots of e-commercial website, such as Amazon, Airbnb and Netflix. Some websites release their rating datasets for non-commercial purpose, encouraging the computer science community to develop more accurate systems (Funk 2006). There are two common methods to produce recommendations, collaborative filtering and content-based filtering. Collaborative filtering systems make predictions based on the user's past behavior as well as similar decisions made by others. Content-based filtering systems make predictions based on the characteristic on the items (Burke 2002).

In this project, we used the 1 million dataset collected by Grouplens. We built several recommendation systems, which combined the two techniques described above, and compared their performances in terms of generalization error and computational efficiency. Given a user information, we aimed to predict the user's rating, in a scale of 1-5, on a movie and decided whether to make a recommendation based on the rating.

Previous researches mainly used models such as global average, KNN and SVD on Netflix Prize Data Set (Bennett et al. 2007), which has a rating file similar to our dataset. Therefore, we implemented these models and used them as baseline comparison models. However, Netflix Data Set does not contain additional user and movie information, so our project cooperated this information provided by Movielens, and implemented some variants of existing models for recommendation systems, as well as other models such as XGBoost, and compared their performances in terms of accuracy and time complexity.

II. BACKGROUND

The dataset is collected by Grouplens, a computer research lab in University of Minnesota, Twin Cities. The dataset includes a rating file, which contains 1,000,209 ratings of 3,952 movies from 6,041 Movielens users. Each user rated at least 20 movies, and all the ratings range from 1 (worst) to 5 (best). Each row is a rating entry. The first column is user ID, the second column is movie ID, and the third column is the rating. The fourth column is the corresponding timestamp.

In addition to the rating file, there is information about users and movies in "users.dat" and "movies.dat" respectively. In "users.dat" file, each row is the information of a user specified by user ID. There are 6,041 users, so there are 6,041 rows. The first column is user ID, the second column is gender, the third column is his or her age, the fourth column is the occupation, and the fifth column is the zip code.

In "movies.dat" file, each row is the information of a movie specified by movie ID. There are 3,952 users, so there are 3,952 rows. The first column is movie ID, the second column is movie title, and the third column is its genre (Action, Comedy, etc.).

III. MODELS

A. K Nearest Neighbor (KNN)

The first approach we tried was K Nearest Neighbor (KNN). We assumed that similar user will rate similar movies similarly. Given a point $(u,\ m)$, we compared it with all other points and select the most similar K points. We computed the rating by averaging the ratings of these K points. We improved the KNN performance by using weighted averaging technique and by applying different similarity metrics. We chose the best K by cross validation.

1) Item-based KNN

The item-based KNN follows the intuition that one user will prefer similar movies. We predicted the rating of user u on item i by taking the average of top K items that are similar to item i. We calculated the item similarities by between item i and j using cosine similarity:

$$sim(i,j) = \frac{r_i \bullet r_j}{||r_i||||r_j||}$$

Suppose we choose the top K similar items with ratings $(r_1,\ r_2, ...,\ r_k)$ and similarities $(sim(i,1),\ sim(i,2), ..., sim(i,k))$. We predicted the rating of user u on item i to be,

$$\widehat{r_{ui}} = \frac{\sum_{j=1}^{k} r_j sim(i,j)}{\sum_{j=1}^{k} sim(i,j)}$$

2) User-based KNN

With the same logic, we can use the user similarity to predict the rating. Since user-based data is less sparser than movie-based data, we are able to use Pearson correlation coefficient and adjusted cosine similarity as the similarity metrics.

In statistics, Pearson correlation coefficient is a measure of the linear correlation between variables *X and Y*. It ranges from -1 to 1. We computed the similarity between user u and v to be,

$$sim(u,v) = \frac{\sum_{i \in I}(r_{ui}-\overline{r_u})(r_{vi}-\overline{r_v})}{\sqrt{\sum_{i \in I}(r_{ui}-\overline{r_u})^2 (r_{vi}-\overline{r_v})^2}}$$

I is the set of common items that both user u and v have rated. $\overline{r_u}$ is the average rating of user u across rated items and $\overline{r_v}$ is the average rating of user v across rated items.

We predicted the rating of user u on item i to be,

$$\widehat{r_{ui}} = \frac{\sum_{v=1}^{k} sim(u,v)(r_{vi}-\overline{r_k})}{\sum_{v=1}^{k} abs(sim(u,v))} + \overline{r_u}$$

This formula indicates that with higher correlation (close to 1), when user v rates item i high (relative to his mean), user u will have similar rating behavior.

With less sparser data, Pearson Correlation Coefficient generally performs better than cosine similarity with one disadvantage that it doesn't consider the difference of rating scales of different items. We overcome this drawback by using adjusted cosine similarity. The similarity between user u and v now becomes,

$$sim(u,v) = \frac{\sum_{i \in I}(r_{ui}-\overline{r_i})(r_{vi}-\overline{r_i})}{\sqrt{\sum_{i \in I}(r_{ui}-\overline{r_i})^2 (r_{vi}-\overline{r_i})^2}}$$

where $\overline{r_i}$ is the average rating of item i.

We predicted the rating using the same technique as in Pearson Correlation Coefficient user-based KNN.

B. Singular Vector Decomposition (SVD)

Besides KNN, we tried the latent factor model by using SVD (Singular Vector Decomposition) as a regional effect and combining its predicted value with global effect such as user and movie biases. This turned out to be much better than KNN model in terms of RMSE.

Latent factor model tries to explain the ratings by characterizing both movies and users on latent patterns (Koren et al. 2009). SVD is one of the most commonly used matrix factorization methods, which realizes the latent factor model. This model maps both users and items to a joint latent factor space, with f dimensions (number of latent features). The key formula for SVD is $M = U\Sigma V^*$. M is the rating matrix. Each row is a movie, and each column is a user. Each element of M is the corresponding rating. By SVD, it can be factorized into three matrices. U is the movie-feature matrix with vectors q_i, and V^* is the user-feature matrix with vectors p_u. Each movie i is associated with a vector $q_i \in R^f$, and each user u is associated with a vector $p_u \in R^f$. The element in those vectors measures the extent to which the item possesses those factors. Σ is the diagonal matrix with the values of those "extents" on the diagonal as eigenvalues. In practice, we use low-rank matrix approximation, which only choose top r features with largest singular values.

The resulting dot product of q_i transpose and p_u leads to the rating of user u on movie i, so we can estimate and update those vectors to approximate the true rating matrix. We used gradient descent to estimate the unknown ratings stepwisely by iterations.

1) Brute-force SVD

We first tried brute force SVD. Since SVD is only well-defined for matrices with no missing values, we imputed all missing values with global average. Then we decomposed the rating matrix into $U\Sigma V^*$, and chose top 50 latent features with largest eigenvalues. This simple SVD outperformed KNN method with a lower testing RMSE. This method served as a baseline model for our variants of SVD models.

2) SVD + Bias

We also combined SVD with global effects such as user and movie biases. Those global effects were served to measure the overall deviations from predicted ratings by SVD and global average. To be more specific, our new predicted formula is given by

$$\widehat{r_{ui}} = \mu + b_i + b_u + q_i^T p_u$$

In this way, one single rating is broken into four parts: global average μ, movie bias b_i, user bias b_u and used-item interaction, which is recovered by SVD. We used gradient descent to approximate the true rating matrix by 30 iterations. In each step, the system learns by minimizing the square error function:

$$min \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + {b_u}^2 + {b_i}^2)$$

where λ denoted the l-2 regularization term, obtained by cross validation.

Due to high volumes of missing values in the rating data, we will also try some modifications such as SVD++ (Koren et al. 2008), which added additional terms to indicated if the movie was rated before.

C. XGBoost

In addition to the ratings, MovieLens also provides user profile and movie profiles. Therefore, we used this additional information to build XGBoost model. Here we used different preprocessing techniques than before, since the data we used is different. There is an existing XGBoost library, so we focused more on data preprocessing and turning parameters.

1) Preprocessing

First, we examined each attributes in the user and movie information files. We determined their meanings, encoded into the correct format, and imputed missing values based on their specific situations. For example, in "Age" attribute from user information, each number (such as 1 or 20) does not indicate the specific age, but a group of age ranges instead. Thus, we created a dummy variable for each age group and imputed the missing values with the most frequent group.

All attributes were treated as categorical variables, and missing values were imputed by mode. We found most of the encodings reasonable, but one attribute, zip code of users. Similar researches always ignore this attribute, and very few studies addressed the problem to utilize zip code as the additional information. Nevertheless, we found this attribute important, since the plot of average ratings by different location groups and the pairwise mean test suggested that average ratings varied significantly among some potions of groups. Thus, we tried to create dummy variables based on the first one or first two numbers in the zip code. But the result may suggested smarter ways to deal with this attribute, as discussed later.

2) Tuning Parameter

We used grid search and crossing validation to decide the optimal parameters for our XGBoost model. Since the running time for XGBoost takes long time on our dataset, we used a subsample each time to improve the efficiency.

IV. EMPIRICAL RESULT

1) Baseline: Average Measurement

We used different average as the baseline and compared the model results to the baseline results. Global average is computed as the average across all ratings. Assume the user distribution is independent, we estimated the rating of rating of user u on item i to be the average rating of user u. Similarly, we used the movie average as the predicted ratings.

A more accurate way to make a prediction is to take user-specific effect and movie-specific effect into account. If a user is critical, the user is likely to give low rate constantly. The user-item average is calculated as,

$$\widehat{r_{ui}} = \mu + b_i + b_u$$

where μ is the global average and b_i, b_u are the movie bias, the user bias respectively.

	RMSE
Global Average	1.11624
User Average	1.0345
Movie Average	0.9802
User-Item Average	0.9363

For other methods we tried, the results of testing RMSE are listed below.

2) KNN

	RMSE
Item-Based KNN (Cosine Similarity)	1.11624
User-Based KNN (cosine similarity)	1.0345
User-Based KNN (Pearson Correlation)	0.9802
User-Based KNN (Adjusted Cosine Similarity)	0.9362

3) SVD

	RMSE
brute-force SVD	0.9626
SVD+Bias	0.8875
SVD ++	0.8985

4) XGBoost

	RMSE
best case	1.0297
average	1.0729
worst case	1.1193

V. DISCUSSION

As shown in the result tables, it turned out that the models based on SVD still performed better than other models, since the average generalization errors are much smaller. For KNN and XGBoost, once we found out the best way to turn parameters and dummy code, they performed slightly better than simply using averages to predict.

On one hand, this suggested the usefulness of combining global and regional effects, as used in SVD models. On the other hand, a smarter ways to preprocess data, especially the user and movie profiles can improve model performances.

We also tried ensemble models, which find the best linear combinations from the predicting results of KNN, SVD and XGBoost, but the result is not significantly better than baseline averages.

Another thing to note is the efficiency. Our SVD models took much less time than others, with an average of 10 minutes. However, KNN, which has the highest time complexity, took 14 hours to run on the whole data set. So in practical recommendation systems, we would favor SVD models and their variants.

VI. FUTURE WORK

Although SVD turned out to be our best model with minimum RMSE, future work is still need to find smarter ways to utilize other information such as user information and movie information and reduce missing values. In particular, using movie or user similarity based on this information to predict missing ratings can help reduce missing values, but it turned out difficult to find a good way to find the appropriate neighbors.

In addition, some domain knowledge is necessary for the purpose of data preprocessing. For example, understanding what the zip code indicates about user's geographic information may help with dividing users into appropriate groups and then create smarter dummy variables for those groups. Very few researches have been done to encode these geographic information and add spatial-temporal model to recommendation system. Therefore, those information can be better utilized in the future.

It seems that even combining global and regional effects, the RMSE is bounded by certain threshold and converged slowly around it. Thus we could examine more about the reason behind it in the future.

VII. REFERENCE

Bennett, James, and Stan Lanning. "The netflix prize." Proceedings of KDD cup and workshop. Vol. 2007. 2007.

Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12.4 (2002): 331-370.

Funk, Simon. "Netflix update: Try this at home." (2006).

Hong, Ted, and Dimitris Tsamis. "Use of knn for the netflix prize." CS229 Projects (2006).

Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 42.8 (2009).

Xue, Gui-Rong, et al. "Scalable collaborative filtering using cluster-based smoothing." *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005.