Recommender system using collaborative filtering

z5150150 Chenghao Ma | z5142340 Haiyu LYU | z5143887 Li Zhou | z5151800 Siwei Lin

1. Introduction

The Collaborative filtering (CF) algorithm attempts to find a model that provides a specific suggestion through user behavior. He judges their preference for the movie based on the user's behavioral data, such as the user's rating of the movie they have watched. The key issue is that if two users have similar ratings for other items, then one of the users will not rate the product. The rating should be similar to another user. Due to the simplicity and effectiveness of the method based on the neighborhood (user, item), they are able to produce accurate personalized recommendations, but are limited by size because of the need to calculate similarities between users or between commodities, in large-scale data. The user's rating is sparse, with only a small percentage of users or products rated.

User-based collaborative filtering and item-based collaborative filtering systems are called Memory based collaborative filtering technologies. Their common shortcoming is that data is sparse and difficult. Processing large amounts of data affects immediate results, so a model-based collaborative filtering technique has been developed.

The model-based method uses scoring to learn an effective model, and uses the machine learning method to learn the user-item interaction relationship to obtain a model. It is said that the model-based collaborative filtering method is a better recommendation system method for establishing collaborative filtering. There are many machine learning methods that can be used to build models. The experiment uses matrix factorisation, singular value decomposition (SVD), which decomposes goods and users into spaces that can represent user-item recessive relationships. Behind it is how the recessive feature represents the user's rating of the product, so we can estimate the rating of the user without rating the product.

2. Method Principle

2.1 K Nearest Neighbours (KNN)

The core idea of the kNN algorithm is that if the majority of the most neighboring samples of a sample in the feature space belong to a certain category, the sample also belongs to this category and has the characteristics of the samples on this category. In determining the classification decision, the method only determines the category to which the sample to be divided belongs according to the category of the nearest one or several samples. The kNN method is only related to a very small number of adjacent samples in the category decision. Since the kNN method mainly relies on the surrounding limited samples, rather than the method of discriminating the class domain to determine the category, the k-nearest neighbor method is better than other methods for the cross-over or overlapping sample set of the class domain.

KNN is a machine learning algorithm used to find common inter-reader clustering based on common movies rating, and can be predicted based on the average score of the nearest k neighbors. And the principle of user_based KNN:

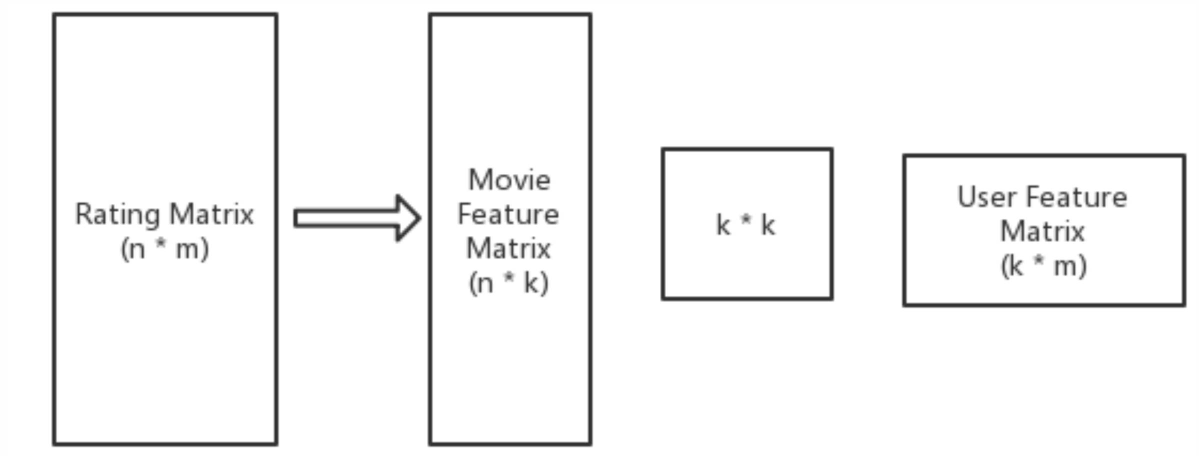
$$\hat{x}_{ui} = \frac{\sum_{u'} sim_u(u,u') * r_{u'i}}{\sum_{u'} \|sim_{u'}(u,u')\|}$$

2.2 Singular Value Decomposition (SVD)

SVD, known as singular value decomposition, is a matrix decomposition technique in linear algebra. It can decompose any matrix of n x m into(M, ∑, U) ,M is An matrix of n x k, ∑ is an orthogonal matrix of k x k, U is a matrix of k x m, and A=M∑U. After the matrix A is decomposed by the SVD method, if only the first k largest singular values are retained, the purpose of dimension reduction of the matrix is achieved.

SVD decomposes the original dataset matrix Data into three matrices U, ∑, V The original matrix Data n*m is n rows and m columns, M_{n*k} represents n rows and k columns **∑k∗k** means k rows and k columns U_{k*m} represents k rows and m columns.

$$Data_{n*m} = M_{n*k} * \sum(k*k) * U_{k*m}$$



The evaluation indicator metric uses RMSE. We define the loss function as follows:

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(V_{ij} - p(U_i, M_j))^2 + \frac{k_u}{2}\sum_{i=1}^{n}\|U_i\|^2 + \frac{k_m}{2}\sum_{j=1}^{m}\|M_j\|^2$$

The classical SVD algorithm, the calculation formula for predicting the score is as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^2 p_u$$

Where μ is the average of the scores, which represents the offset amount of the u user and the offset amount of the i item, respectively.

$$\sum_{r_{ui} \in \mathbb{R}_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

Thus, the number of variables needs to be optimized by stochastic gradient descent is four and the expressions need to be optimized is of the form:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$

$p_u \leftarrow p_u + \gamma(e_{ui}q_i - \lambda p_u)$

$q_i \leftarrow q_i + \gamma(e_{ui}q_u - \lambda q_i)$

The smaller the SSE, the smaller the overall loss and the more accurate the prediction.

2.3 Bayesian Personalized Randking (BPR)

BPR method provide a user with a ranked list of items.The U is the set of all users and I is the set of all items, the recommender system is now to provide the user with persionalized total ranking of all items.

The method use different methods and project pairs as training data, and optimize the correct ranking of item pairs instead of scoring individual items, from S we try to rebuild each user part. If user u has viewed an item, then it could be assumed that the user prefers this item instead of all other unobserved items. For formalization, we create training data $D_S = U * I * I$ by:

$D_S := (u, i, j)|I_u^+ \wedge j \in I \setminus I_u^+$

This is a general approach to solving personalized ranking tasks. It consists of a general optimization criterion for personalized rankings, BPR-OPT, which will be derived from Bayesian using the likelihood function to analyze the problem for $p(i > j|\theta)$ and the model's prior probability parameter $p(\theta)$. We show the analogy of the ranking statistics AUC. Through BPR optimization, these models are able to produce a better training method than ranking using BPR.

The generic optimization criterion for personalized ranking BPR_OPT is:

$BPR - OPT := ln\ p(\theta| >_u) = \sum_{(u,i,j)\in D_s} ln\sigma(\hat{x}_{uij} - \lambda_\theta \|\theta\|^2)$

2.4 Similarity calculation

2.4.1 Cosine similarity

Calculate the cosine of the two vectors:

$cos(u_k, u_a) = \frac{u_k * u_a}{\|u_k\| * \|u_a\|} = \frac{\sum x_{k,m}x_{a,m}}{\sqrt{\sum x_{k,m}^2 x_{a,m}^2}}$

If the angle is 90°, the similarity is 0. If the angle is the same, the similarity is 1. it can also be normalized by 0.5 + 0.5 ∗ cos θ.

2.4.2 Euclidean distance similarity

Similarity $= \frac{1}{1+norm(A-B)} = \frac{\sum x_{k,m}x_{a,m}}{\sqrt{\sum x_{k,m}^2} * \sqrt{\sum x_{a,m}^2}}$

The smaller the Euclidean distance, the greater the similarity between the two users. The larger the Euclidean distance, the smaller the similarity between the two users.

3. Implementation

Our project is implemented by **python 3.7**.

Besides, there are some functions to calculate cosine similarity and Euclidean distance similarity.

```
# Cosine similarity
def get_cos_similarity(self, ratings):
  return cosine_similarity(ratings)
```

```
# Euclidean distance similarity
def get_similarity_euc(self, ratings):
        return euclidean_distances(ratings)
```

3.1 K Nearest Neighbours (KNN)

Based on user-based collaborative filtering, it finds k user groups that are similar to a certain user. Then recommend this movie of interest to this group to this user.

First, a similarity matrix is constructed, which is based on calculating the distance between two users. Here, we use the calculated cosine similarity and Euclidean distance similarity to measure the distance between them.

Next, we need to calculate the k users most relevant to a certain user. Then, the average score of the k users is used for prediction. The following algorithm is based on this formula:

$$\hat{x}_{ui} = \frac{\sum_{u'} sim_u(u,u') * r_{u'i}}{\sum_{u'} \|sim_{u'}(u,u')\|}$$

```
def predict_topk(self, ratings, similarity, k=10):
  pred as a numpy matrix with zero tuples
  for i in all ratings.shape[0]:
    get the top_k_users sorted with similarity
    for j in all ratings.shape[1]:
      pred is equal to dot similarity with ratings
      pred is equal to pred divided with the abs of similarity
```

Finally, we only need to adjust the parameter k so that we can find the best rmse.

```
for k in all k_array:
  get the top k user_pred
  find the user train rmse and user test rmse
```

3.2 Singular Value Decomposition (SVD)

For SVD function, we have a class called SVD(object), which can be used to train, test and evaluate the SVD model. We have four matrics: bu, bi, p and q. The values in them are initialized using normal distribution and upgraded in stochastic gradient descent algorithm with learning rate is 0.01. We have set the number of trainings to be 50 to find out the most optimal value. Because this model might be overfitting if we have trained it many times.

Some hyperparameters are as following:

```
self.num_factors = 600 # Dimension of the Latent Factor
self.regs = 1e-3       # Regularizer Coefficient
self.lr = 0.01         # Learning Rate
self.trains = 50       # How many number of Training Loops
self.batch_size = 228  # How many data is fed in each training
```

We build four matrix which are bu, bi, p and q, and then initialize all the parameters (Use Normal Distribution):

```
bu is equal to Gaussian Distribution of same size with num_user
bi is equal to Gaussian Distribution of same size with num_item
p is equal to Gaussian Distribution of same size with
tuple(num_user,num_factors)
q is equal to Gaussian Distribution of same size with
tuple(num_factors,num_item)
```

The classical SVD algorithm, we predict the score, where μ is the average of the scores, which represents the offset amount of the u user and the offset amount of the i item, respectively. And the number of variables needs to be optimized by stochastic gradient descent is four:

```
bu = bu + lr ∗ (error − regs ∗ bu)
bi = bi + lr ∗ (error − regs ∗ bi)
p  = p  + lr ∗ (error ∗ q − regs ∗ p)
q  = q  + lr ∗ (error ∗ p − regs ∗ q)
```

3.3 Bayesian Personalized Randking (BPR)

We have a class BPR(object), which can be used to train, test and evaluate the BPR model. This method is different from the other two methods, so we use a seperate function load_data() to load training data and testing data. This method only uses two columns: USER and ITEM, and it will make a binary classification. So, we cannot use rmse as the evaluation score, we choose AUC (Area Under the Curve) instead. The AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example.

Some hyperparameters are as following:

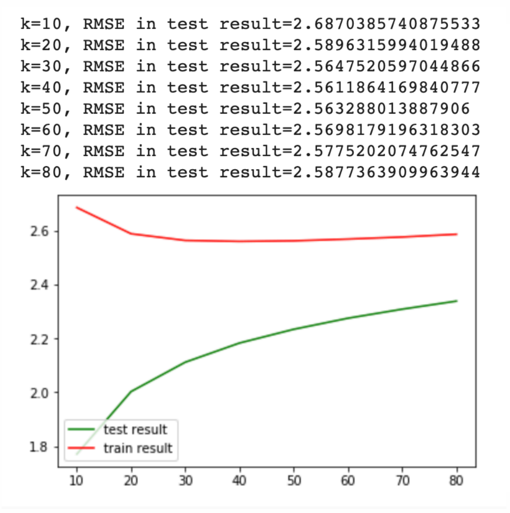
```
self.num_factors = 600 # Dimension of the Latent Factor
self.reg = 0.01        # Regularizer Coefficient
self.lr = 0.05         # Learning Rate
self.train_count = 300 # How many number of Training Loops
```

```
U  += −lr ∗ (−mid ∗ Vi − Vj) + reg ∗ U
Vi += −lr ∗ (−mid ∗ temp) + reg ∗ Vi
Vj += −lr ∗ (−mid ∗ (−temp)) + reg ∗ Vj
```

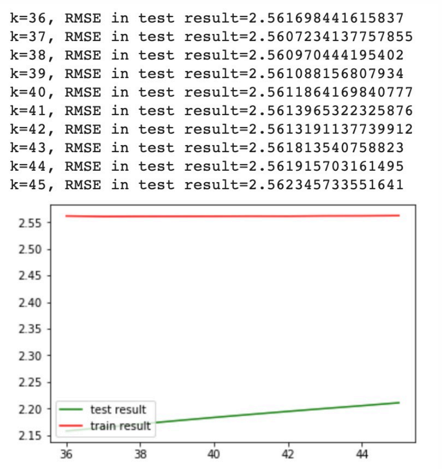
4. Results

4.1 K Nearest Neighbours (KNN) results

KNN with Cosine similarity:



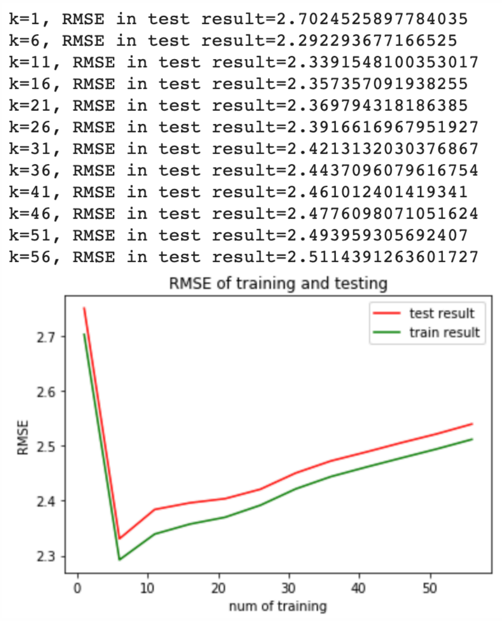
```
k=10, RMSE in test result=2.6870385740875533
k=20, RMSE in test result=2.5896315994019488
k=30, RMSE in test result=2.5647520597044866
k=40, RMSE in test result=2.5611864169840777
k=50, RMSE in test result=2.563288013887906
k=60, RMSE in test result=2.5698179196318303
k=70, RMSE in test result=2.5775202074762547
k=80, RMSE in test result=2.5877363909963944
```



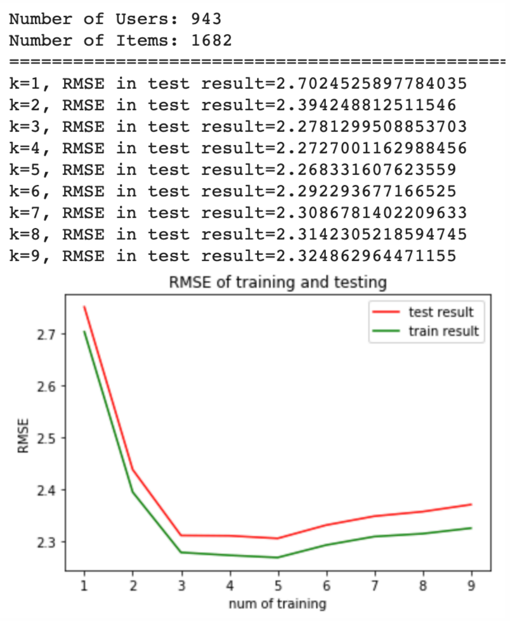
```
k=36, RMSE in test result=2.561698441615837
k=37, RMSE in test result=2.5607234137757855
k=38, RMSE in test result=2.560970444195402
k=39, RMSE in test result=2.561088156807934
k=40, RMSE in test result=2.5611864169840777
k=41, RMSE in test result=2.5613965322325876
k=42, RMSE in test result=2.5613191137739912
k=43, RMSE in test result=2.561813540758823
k=44, RMSE in test result=2.561915703161495
k=45, RMSE in test result=2.562345733551641
```

From the figure above, the result of K Nearest Neighbours with Cosine similarity shows that k equals to about 37 which performs better. And with the increasing of k's value, the RMSE of training data is steady rise, however, the RMSE of testing data is steady decreasing and the best result is the RMSE remained steady at about 2.56072.

KNN with Euclidean Distance:



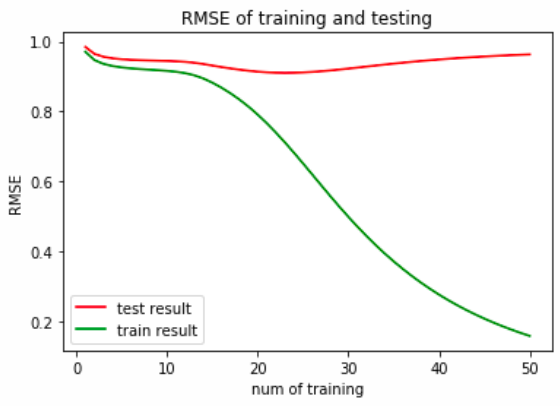
```
k=1, RMSE in test result=2.7024525897784035
k=6, RMSE in test result=2.292293677166525
k=11, RMSE in test result=2.3391548100353017
k=16, RMSE in test result=2.357357091938255
k=21, RMSE in test result=2.369794318186385
k=26, RMSE in test result=2.3916616967951927
k=31, RMSE in test result=2.4213132030376867
k=36, RMSE in test result=2.4437096079616754
k=41, RMSE in test result=2.461012401419341
k=46, RMSE in test result=2.4776098071051624
k=51, RMSE in test result=2.493959305692407
k=56, RMSE in test result=2.5114391263601727
```



```
Number of Users: 943
Number of Items: 1682
==============================================
k=1, RMSE in test result=2.7024525897784035
k=2, RMSE in test result=2.394248812511546
k=3, RMSE in test result=2.2781299508853703
k=4, RMSE in test result=2.2727001162988456
k=5, RMSE in test result=2.268331607623559
k=6, RMSE in test result=2.292293677166525
k=7, RMSE in test result=2.3086781402209633
k=8, RMSE in test result=2.3142305218594745
k=9, RMSE in test result=2.324862964471155
```

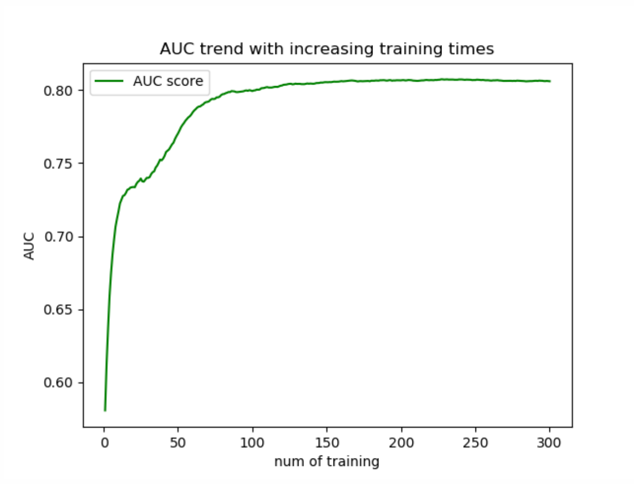
From the figure above, the result of K Nearest Neighbours with Euclidean Distance similarity shows that k equals to about 5 which performs better. After that, with the increasing of k's value, the RMSE of training data is steady rise, as well as the RMSE of testing data is also steady increasing and the best result is the RMSE remained steady at about 2.26733.

4.2 Singular Value Decomposition (SVD)



From the figure above, the result of Singular Value Decomposition shows that during the increasing of the number of training data, the RMSE of training result is decrasing dramatically, however the test result steady at about 0.9. Additionally, it performs better than KNN results.

4.3 Bayesian Personalized Randking (BPR)



From the figure above, the result of Bayesian Personalized Randking shows that the Area under hte Curve (AUC) is almost stability after the number of training data up to 150 and the AUC score remain steady at about 0.807.

5. Discussion

	KNN with Euclidean	KNN with Cosine Similarity	Singular Value Decomposition (SVD)	Bayesian Personalized Randking (BPR)
RMSE	2.26733	2.56072	0.9	-
AUC	-	-	-	0.807

For the results of K Nearest Neighbours with cosine similarity and Euclidean distance similarity as well as Singular Value Decomposition shows above, the performance of Singular Value Decomposition is better which lowest RMSE is 0.9.

As for Bayesian Personalized Randking, it predicts a binary result rather than ratings, so it is hard to compare with other two algorithms and we use AUC to evaluate the model. In the result, we found the AUC value is bigger than 0.8, according to the guide document, this model is good to make recommendation.

6. Conclusion

The SVD is used in the recommendation system, the recommendation result is relatively accurate, and the model is also very expandable and can be applied to various scenarios. Overall, SVD has good prospects for the application of recommended systems. Even though SVD and kNN are designed for the item prediction task of personalized ranking from implicit feedback, none of them is directly optimized for ranking.

So, we have tried another popular recommender system: Bayesian Personalized Randking. BPR is a sorting algorithm based on matrix decomposition, but compared with algorithms such as SVD, it is not a global scoring optimization, but a sorting optimization for each user's own commodity preferences. Therefore, the idea of iterative optimization is different. At the same time, the requirements for the training set are also different.

Reference

1. Ma C C. A Guide to Singular Value Decomposition for Collaborative Filtering[J]. 2008.
2. Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems[J]. Computer, 2009, 42(8):30-37.
3. Steffen, R., Christoph, F., Zeno, G. and lars, ST. (2009). *BPR: Bayesian Personalized Ranking from Implicit Feedback*. Machine Learning Lab, University of Hildesheim, Marienburger Platz 22, 31141 Hildesheim, Germany.
4. A. Paterek, "Improving Regularized Singular Value Decomposition for Collaborative Filtering," *Proc. KDD Cup and Workshop*, ACM Press, 2007, pp. 39-42.
5. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In ICML'05: Proceedings of the 22nd international conference on Machine learning, pages 89–96, New York, NY, USA, 2005. ACM Press.
6. Rosenthal, E. (2019). *Intro to Recommender Systems: Collaborative Filtering | Ethan Rosenthal*. [online] Ethanrosenthal.com. Available at: https://www.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/ [Accessed 12 Jul. 2019].
7. Lettier, D. (2019). *k-Nearest Neighbors from Scratch by David Lettier*. [online] Lettier. Available at: https://lettier.github.io/posts/2016-06-10-k-nearest-neighbors-from-scratch.html [Accessed 12 Jul. 2019].
8. Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L., 2009, June. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (pp. 452-461). AUAI Press.
9. Fawcett, T., 2006. An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), pp.861-874.