FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Algorithmische Grundlagen des Maschinellen Lernens LAB

Project

Recommender System

Chengcheng Guo 183090
Tutor : Paul Kahlmeyer

Jena, July 11, 2021

Contents

1 Introduction

A recommender system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item[1]. The aim of this project is to implement a recommender system with Python and calculate the train and test error.

2 Data

The given dataset is an array of size 380311 $\times$ 3. The 3 columns are userID, itemId and Rating. The largest userID is 5498, the largest itemID is 2079, rating is from 0 - 4(Figure 2.1).

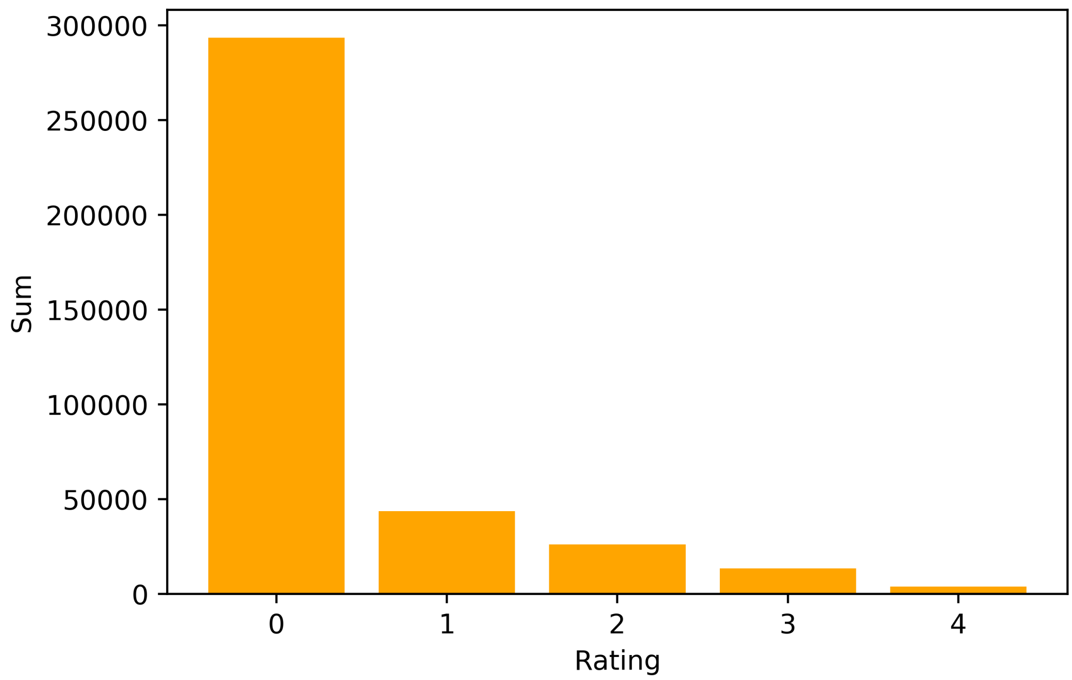

Figure 2.1: Distribution of ratings

To separate rate = 0 and no rate, firstly we add 1 to all known ratings, therefore the range of the ratings turn into 1 to 5. Then we split the dataset into 2 parts: 80% for training and 20% for testing. The train and test dataset can be converted into a 5499 $\times$ 2080 sparse matrix separately(Users $\times$ Items).

3 Methods

3.1 Baseline Predictors

Baseline prediction method is useful for pre-processing and normalizing data for use with more sophisticated algorithms. The simplest baseline is to predict the average rating over all ratings in the system:

$$\hat{r}_{ui} = \mu \tag{3.1}$$

where μ is the overall average rating. Baselines can be further enhanced by combining the user mean with the average deviation from user mean rating for a particular item. Therefore we use a baseline predictor of the following form[2]:

$$\hat{r}_{ui} = \mu + b_u + b_i \tag{3.2}$$

where $b_{u,i}$ is baseline prediction for user u and item i, μ is the overall average rating, b_u and b_i are user and item baseline predictors, respectively. They can be defined simply by using average offsets as follows:

$$\begin{aligned} b_u &= \mu_u - \mu \\ b_i &= \mu_i - \mu \end{aligned} \tag{3.3}$$

where μ_u is (arithmetic) average of all ratings given by user u, and μ_i is (arithmetic) mean of all given ratings for item i.

3.2 KNN: K-Nearest Neighbors algorithm

KNN algorithm is called K nearest neighbor classification algorithm. The core idea of the KNN algorithm is: if the majority of the k most similar neighbors of sample in the feature space belongs to a certain category, then the sample is considered to belong to this category. We will calculate prediction score of user u for i as following steps:[3]
Step 1: Generate a two-dimensional user-item matrix with scores $R_{m\times n}$, where each score is $r_{u,i}$.
Step 2: Calculate the similarity between each 2 user using Pearson's correlation similarity as sim(u,u'). Generate a user similarity matrix.
Step 3: Based on the results obtained in Step 2, find the K scores with the highest weight, the corresponding K users are the neighbours of u.
Step 4: Use formula 3 to calculate the predicted value of i for the target user u.

In this way, we can calculate the predicted scores of target users for movies with no scores, and the N movies with the highest scores can be recommended to user.

In **Step 2**, the similarity between 2 users is calculated by Pearson's correlation. The Pearson coefficient is computed between the target user and all the other users. Pearson correlation coefficient between the users i and j is defined as follows:

$$\operatorname{sim}(i,j) \quad = \quad \frac{\sum_{u\in \text{InJ}} (r_{ui}-\mu_i)(r_{uj}-\mu_j)}{\sqrt{\sum_{u\in \text{InJ}} (r_{ui}-\mu_i)^2}\sqrt{\sum_{u\in I\cap J} (r_{uj}-\mu_j)^2}} \tag{3.4}$$

where r_{ui} is rating Item i from User u, r_{uj} is rating Item j from User u. μ_i, μ_j are mean of all ratings given for item i and j respectively. U_i, U_j set of all users who have submitted a rating for item i and j respectively. The similarity that cannot determine the similarity is specified as "inf". We save the similarity matrix in sim_train_2.csv.

Next, we use KNN to predict the ratings for each pair (user, item) with different k (from k = 1 to k = 35).

4 Error of two methods

We use RMSE and MSE to determine the prediction error. The error of baseline method and KNN method is listed in table: 4.1.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(r_{ui} - \hat{r}_{ui}\right)^2 \tag{4.5}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(r_{ui} - \hat{r}_{ui}\right)^2} \tag{4.6}$$

where n is number of ratings in the test data, $\hat{r}_{u,i}$ is predicted rating for user u, item i,$r_{u,i}$ is the real rating for user u, item i.

The train and test error can be seen in Figure 4.2 and Figure 4.3.

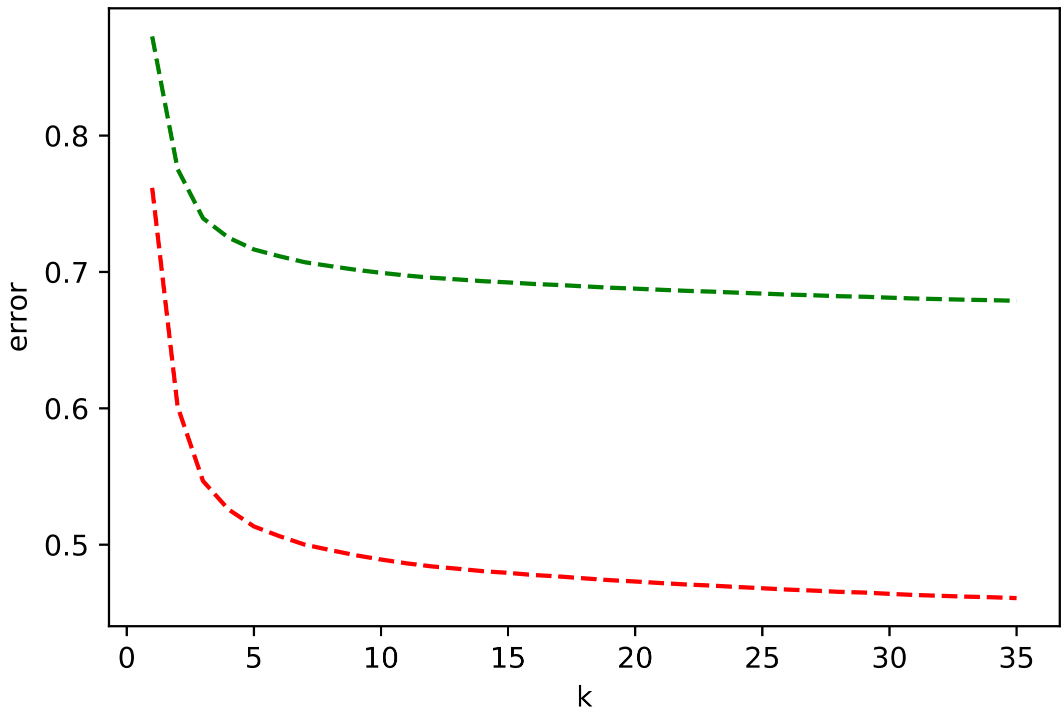

Figure 4.2: train Error with different k

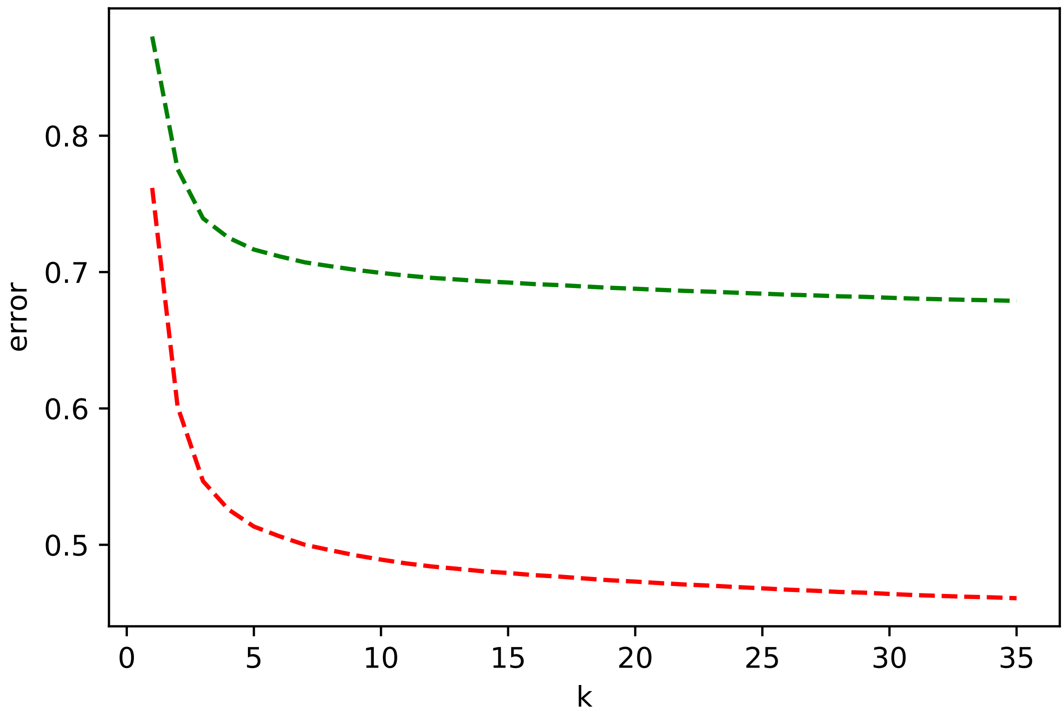

Figure 4.3: test Error with different k

Method	Baseline train	Baseline test	KNN(k = 35) train	KNN(k = 35) test
RMSE	2.09	2.09	0.67	0.68
MSE	4.35	4.35	0.45	0.46

Table 4.1: Train and test error of method of baseline and KNN(k = 35)

References

[1] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*. In: Recommender Systems Handbook (2011), pp. 1–35.

[2] John T. Riedl Michael D. Ekstrand and Joseph A. Konstan. *Collaborative Filtering Recommender Systems*. In: Foundations and Trends in Human–Computer Interaction 4.2 (2010), pp. 81–173.

[3] BeiBei CUI. *Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm*. In: ITM Web of Conferences 12 (2017), pp. 3–5. DOI: 10.1051/itmconf/20171204008.