FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Algorithmische Grundlagen des Maschinellen Lernens LAB

Project

# Recommender System

Chengcheng Guo 183090
Tutor : Paul Kahlmeyer

Jena, July 12, 2021

# Contents

**1 Introduction** **3**

**2 Data** **3**

**3 Methods** **4**
3.1 Baseline Predictors . . . . . 4
3.2 KNN: K-Nearest Neighbors algorithm . . . . . 4

**4 Error of two methods** **5**

# 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 goal of this project is to implement a recommender system with Python to predict ratings of unseen (User,Item) combination then calculate the train and test error.

# 2 Data

The given dataset is an array of 380311 tuples and 3 columns of integers. The 3 attributes are "UserID", "ItemId" and "Rating". The largest UserID is 5498, the largest ItemID is 2079, Rating is in range from 0 to 4, 0 is dislike, 5 is like. Figure 2.1 shows the distribution of ratings. Table 2.1 shows the most popular Items and Items with lowest average Rating.

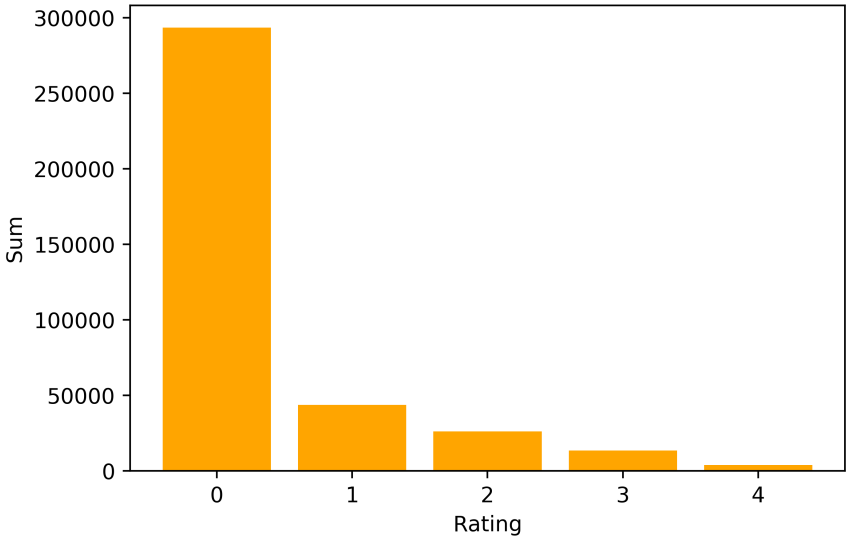


Figure 2.1: Distribution of ratings

| ItemID | average Rating | ItemID | average Rating |
|---|---|---|---|
| 290 | 1.97 | 944 | 0 |
| 292 | 1.75 | 921 | 0 |
| 647 | 1.68 | 908 | 0.0038 |
| 291 | 1.67 | 950 | 0.0043 |
| 536 | 1.66 | 877 | 0.0045 |

Table 2.1: Most popular and lowest rated Items

To separate rate = 0 and no rate, firstly we add 1 to all known ratings, therefore the range of the ratings turns 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 (Users $\times$ Items) sparse matrix respectively .

# 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 $\mu$ 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 $\hat{r}_{ui}$ is baseline prediction for user u and item i, $\mu$ 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 (bias) as follows:

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

where $\mu_u$ is (arithmetic) average of all ratings given by User u, and $\mu_i$ 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.

There are two primary types of neighborhood-based algorithms: User-based collaborative filtering and Item-based collaborative filtering. An important distinction between these two methods is that the ratings in the former case are predicted using the ratings of neighboring users, whereas the ratings in the latter case are predicted using the user's own ratings on neighboring items [3]. We choose to use Item-based methods , for it often provides more relevant recommendations. Due to the fact that a user's own ratings are used to perform the recommendation. Furthermore we have much smaller size of items than users (2080 $<$ 5499), thus we will save running time when generate

a similarity matrix.

Here, we will use the latter method to calculate the prediction score of User u for i according to the following steps:[4]
**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 item using Pearson's correlation similarity as sim(i,j). Generate a item similarity matrix.
**Step 3**: Based on the results obtained in Step 2, find the K items with the highest weight score, the corresponding K items are the neighbours of $Item_i$.
**Step 4**: Use **Step 3** to predict the predicted rating of $Item_i$ for the target $User_u$.

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

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

where $r_{ui}$ is rating Item i from User u, $r_{uj}$ is rating Item j from User u. $\mu_i$, $\mu_j$ 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. I and J stands for set of all users who have given a rating to Item i or j.

If one of the items has no rating or the set U is empty, the similarity cannot be determined. We denote the similarity as "inf". We save the similarity matrix of items in sim_train_2.csv.

In this way, we can calculate the predicted scores of target users for movies with no scores, and the k movies with the highest scores can be recommended to the corresponding user. We will predict the ratings for each pair (user, item) with k in range from 1 to 35.

# 4 Error of two methods

We use Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) to validate predictions.

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

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

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 error of baseline prediction and KNN

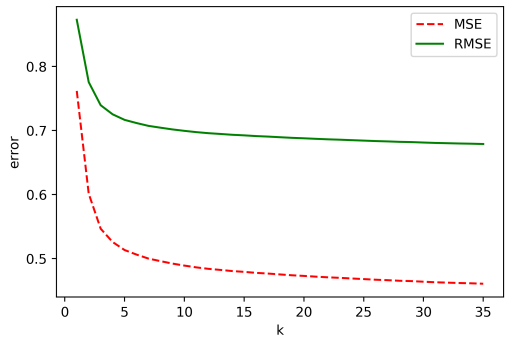


Figure 4.2: train Error with different k

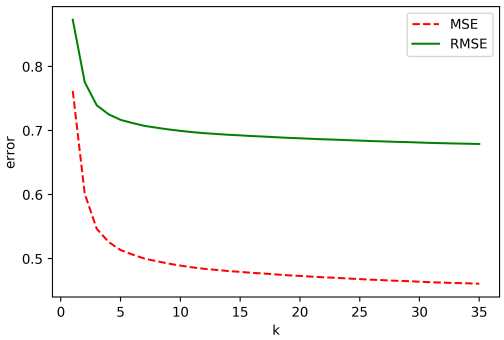


Figure 4.3: test Error with different k

method is listed in table 4.2. The train and test error can be seen in Figure 4.2 and Figure 4.3.

| 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 4.2: Train and test error of method of baseline and KNN(k = 35)

The result indicates that KNN algorithm is more accurate than the baseline predictions. Larger k leads to more accurate predictions (smaller RMSE and MSE).

## 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] Charu C. Aggarwal. *Recommender Systems*. In: Springer (2016), pp. 73–74.

[4] 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.