Project 2- Recommender System

Salya Diallo Tianshuo Zhang Liam Gibbons

I. Introduction

For many people around the world, the use of the internet has now been accompanied by a rise in suggestions for the movies we watch, the music we listen to, and the items we buy. These suggestions are optimized by the use of recommender systems, which is the basis of the following project. In this project, we have to develop several methods and choose the best one to predict a user rating (between 1 and 5) for a given book.

II. Models

We experimented with several different models to improve the Root Mean Squared Error (RMSE) score for the task. Recall that RMSE $= \sqrt{\frac{1}{N}\sum_{r_{ij}}(r_{ij} - p_{ij})^2}$ where N is the number of available ratings r_{ij} and p_{ij} is the predicted rating. A lower RMSE indicates better model performance. Many of the models we tested are based on those introduced during the course, and we will describe them in more detail below.

A. The simplest Model: Linear regression

The first model implemented is one of the simplest prediction model: linear regression (LR). The goal of this method is to estimate the linear relationship between a scalar response (here the rating) and independent variables (here, user, and book id).

Each pair ($user\_id$, $book\_id$) is treated as a feature and their interaction is modeled as a weighted sum to predict the rating:

$$\hat{y} = \beta_0 + \beta_1 \times \text{user\_id} + \beta_2 \times \text{book\_id}$$

where β_0 is the bias term, β_1 and β_2 are weights learned from the data.

The model minimizes the RMSE between the predicted and actual ratings (from the training set).

Given the complexity of user-book interactions, a linear regression approach may be too reductive, potentially overlooking subtle and non-linear connections that influence ratings.

B. Singular Value Decomposition

The second model implemented is matrix factorization using Singular Value Decomposition (SVD) for collaborative filtering, in which both biased and unbiased models can be used and where the learning rate is configurable.

The matrix factorization is performed on the user-item matrix (a matrix where the rows represent users and the columns represent the items) in order to form the latent representations. SVD splits the user-item matrix into three separate matrices: one which represents the users, one which represents the items, and finally the singular matrix where the latent features are represented by the singular values. The matrices representing the users and items have the columns as being representative of the latent features. The latent representations can then be used to calculate missing ratings, by taking the latent representation of both the user and the item and then taking the dot product.

More specifically in our code, we use a SVD class that initializes with many arguments, of which the most important are: the number of latent factors, the number of iterations over the training data, the learning rate, and regularization. The model is then trained using stochastic gradient descent which updates the latent factors (and the biases if chosen). These updates are done by calculating the error obtained from the difference between the actual rating and the model's predicted rating.

C. Using Neural Networks: Neural Collaborative Filtering

A neural collaborative filtering (NCF) [1] model extends the idea of LR by learning nonlinear interactions. Each $user\_id$ and $book\_id$ are represented as a unique embedding vector, which is a learned dense representation capturing latent features of users and books. These embeddings replace traditional features like user or item metadata.

In this model, instead of assuming a simple interaction, NCF learns complex relationships between users and books by concatenating their embeddings and passing them through fully connected neural network layers. This allows the model to capture nonlinear interactions. The architecture is the following:

1) Embedding layer: map $user\_id$ and $book\_id$ to their embedding vectors e_u and e_b respectively.
2) Hidden layer: concatenate e_u and e_b and process them through dense layers with nonlinear activation functions (here, we used ReLU).
3) Output layer: finally, predict the rating as a single value using a regression-like output.

To train this method, we minimized our loss function using the Adam optimizer. The embeddings and the weights are updated simultaneously.

This model is better than linear regression for our problem because it can handle large datasets with many users and books efficiently using embeddings, and it also captures complex nonlinear user-book interactions, improving rating predictions.

Nevertheless, it is more computationally intensive and requires sufficient data for training (sparse datasets may lead to overfitting). The running time of this neural network is above 10 minutes: it is around 27 minutes without a GPU and 17 minutes with Kaggle's GPU.

Note that we had to fine-tune the hyper-parameters of the neural network, which are: the embedding dimension, the

hidden layers dimensions and the learning rate. To obtain our result we used an embedding dimension of 16, hidden layers dimensions of $[64, 32]$ and a learning rate of 0.001.

Finally, it is important to mention that we adjusted the ratings column in our dataset to fall within the range of 0 to 1. This was achieved by normalizing the values based on the minimum and maximum ratings in the training set. Consequently, the predictions generated by the model were also scaled to the range of 0 to 1. To interpret these predictions in their original scale, we transformed them back to the range of 1 to 5.

D. User-Based Collaborative Filtering

User-based collaborative filtering relies on identifying users with similar preferences to the target user to predict the target user's rating for a specific item.

The method starts by creating a user-item matrix that will be used to created a user similarity matrix based on cosine similarity. From those two matrices, the predictions can be obtained by using a variation of the following aggregation function that was introduced in class:

$$r_x(i) = \bar{r}_x + \frac{\sum_{y \in N_U(i)} sim(x,y)(r_y(i) - \bar{r}_y)}{\sum_{y \in N_U(i)} |sim(x,y)|}$$

where $N_U(i)$ is the neighbours of user x and i is the item not yet rated by x. From the user similarity matrix, the weighted sum of the ratings is calculated by only taking into account the ten most similar users to the user in question in order to avoid noise and the subsequent dilution of a more realistic user rating. There is also a mean centering for each user, in order to remove the absoluteness of numbers in the ratings seeing as certain users can be harsher than others.

E. Item-Based Collaborative Filtering

The item-based collaborative filtering method is developed by estimating the ratings through looking at other items that have similar ratings. This method is quite similar to the user-based collaborative filtering method, and therefore many of the steps follow very similar patterns.

Firstly, a matrix is derived from the user-item matrix by calculating the cosine similarity between the items. From there, the two matrices are used to predict the ratings from an adaptation of the following equation presented in class:

$$r_x(i) = \frac{\sum_{j \in N_I(i)} sim(i,j) r_x(j)}{\sum_{j \in N_I(i)} |sim(i,j)|}$$

where $N_I(i)$ is the neighbours of item i, j is the item rated by x. And just like with the user-based collaborative filtering, there is a filtering process of the most similar items in order to improve results.

For both of these methods, it would have been possible to use the Pearson correlation as a similarity metric other than cosine similarity, however with the dataset that we had, it was too computationally expensive and seeing as using cosine similarity gave good results as can be seen in Part 3, we decided to stick on improving the method within the cosine similarity metric.

F. Content-Based Collaborative Filtering

Since we were provided with the ISBNs of each book — which is a unique identifier that can be linked to useful metadata about the book (such as the author, the title, etc.) — another viable approach was content-based filtering. This method involves using an API, such as *Open Library*, to retrieve detailed information about each book in the training and testing datasets. Using this metadata, we could construct a user-item matrix and predict missing ratings. The approach would assign higher ratings to user-book pairs if the book shares similarities in content, author, or genre with books the user has previously rated highly, or if it aligns with preferences of other users who typically enjoy the same books.

Notably, some ISBNs in the dataset were incorrect, requiring preprocessing. Specifically, we identified those not of length 10 and appended 0 to ensure compatibility with the API.

However, we opted not to pursue this approach due to several challenges. These included issues with the APIs considered (Open Library and Google Books), as well as computational constraints. Given the large number of user-item pairs in the dataset, the running time for fetching metadata through the APIs was excessive. Additionally, we were unable to use Kaggle's GPU support for this task, as it involved external API usage, further limiting the feasibility of this approach.

III. Results

LR	NCF	SVD	User-Based	Item-Based
0.90800	0.83587	0.79408	0.83415	0.91627

TABLE I: RMSE of each model obtained on Kaggle

For each of the models that we described above, we obtained a *submission.csv* file with 2 columns (the pair id and the prediction) for them so that we could evaluate their score on Kaggle. As can be seen, our most performant model is the SVD model with a score of 0.79403 and our least performant model is the item-based collaborative filtering model with a score of 0.91627. Moreover, we can remark that NCF and user-based models have a similar performance around 0.83.

Remark that despite its simplicity, the predictions of Linear Regression outperform those of the item-based collaborative filtering model. We can suppose that with additional information, such as book content or author details, this model could perform even better by grouping books based on genuine similarities in their content or authorship.

IV. Conclusion

Looking at the five different models presented above, they cover a variety of different techniques ranging from the use of neural networks to matrix factorization using SVD. These show how there are many different ways to implement a recommender system, which are at the heart of many aspects of today's economy. Of the different recommender systems that we built, the SVD model obtained the lowest RMSE and is therefore the chosen model presented on the Kaggle notebook.

References

[1] Xiangnan He et al. “Neural Collaborative Filtering”. In: (2017), pp. 173–182. URL: https://arxiv.org/pdf/1708.05031.