Project report

Team name: M2 Robo

Lee Chun Yin
3035469140

Chiu Yu Ying
3035477630

Chan Kwan Yin
3035466978
Team leader

1. Introduction

1.1. collaborative filtering (CF)

CF is a technique for recommendation system, in which historical feedback data are used to infer connections between users and products [3]. While additional features can be introduced to offset certain bias effects [1], two inputs (user and product) and one output (user rating on the product) are generally sufficient to train a CF model without involving domain-specific data.

Two major approaches for CF include neighbourhood models and latent factor models. Neighbourhood models compare the similarity between users and recommend products positively rated by similar users, while latent factor models perform dimensional reduction on both users and movies to a common, smaller set of feature attributes such that users are recommended with movies of more coherent features.

1.2. The Netflix Prize dataset

The Netflix Prize is a competition for the prediction of users' favour of movies. The dataset provides existing ratings of users on given movies, and models are trained to predict new ratings.

1.2.1 Dataset format

The dataset contains 100480507 rows of data structured in the following format:

User ID	490189 discrete values
Movie ID	17770 discrete values
Rating	1, 2, 3, 4, 5
Date	Dates from 1999 to 2005

1.2.2 Distribution of ratings

Ratings are mostly distributed around 3 and 4, as shown in Figure 1.

Except for some extreme cases, the number of ratings per user over the 7 years mostly follow an exponential relation for users from 10 to 1000 ratings, as shown in Figure 2. The top 10 users with the highest number of ratings range from 17651 to 8877.

For movies with at least 100 ratings (which is the case for the majority), their numbers of ratings demonstrate a similar but more concave relationship, as shown in Figure 3.

1.2.3 Evaluation process

To evaluate performance, 1425333 rows (about 1.42% of all data) are speified as the standard "probe". We perform analysis in the following procedure:
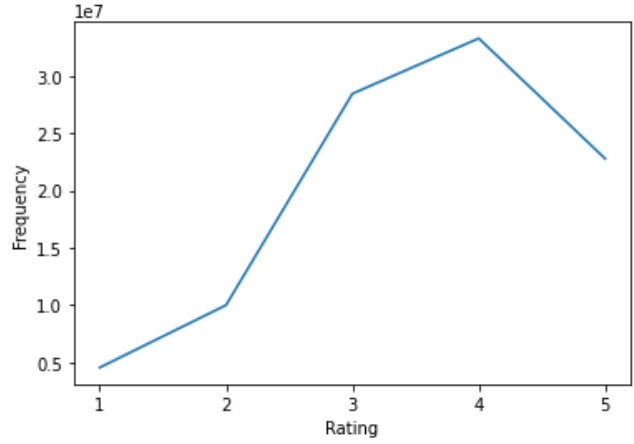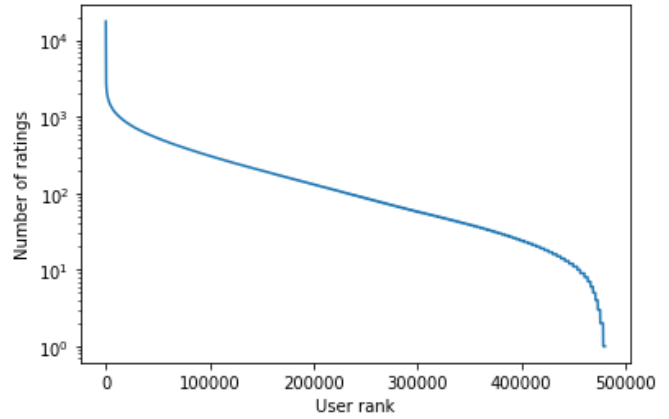

Figure 1. Frequency of ratings
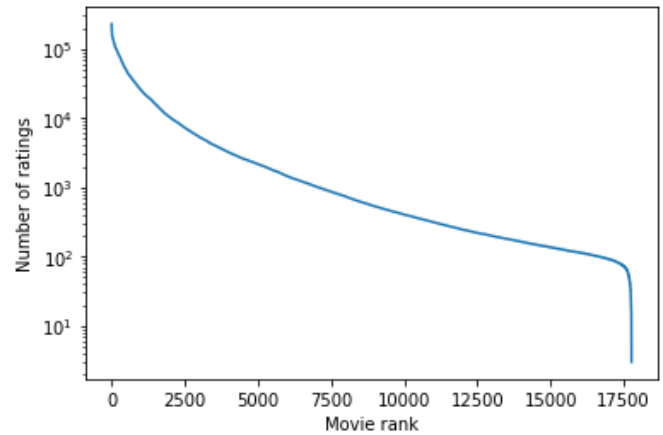

Figure 2. Number of ratings per user

Figure 3. Number of ratings per movie

1. Train the model with the 99055174 non-probe rows.

2. Predict user ratings with the 1425333 probe rows.

3. Compute the root mean squared error (RMSE) be-

tween the predicted data and actual data.

$$\text{RMSE} = \sqrt{\sum_{(i,j) \in E} \frac{(\hat{r}_{ij} - r_{ij})^2}{|E|}}$$

where E is the set of retained evaluation data.

In this project, we evaluate three models, namely:

- k-nearest neighbours (KNN), a simple neighbourhood model

- singular value decomposition (SVD), a latent factor model that accounts for user bias

- neural collaborative filtering (NCF), a latent factor model that represents features with neural network weights

Originally, we proposed three models KNN, SVD and SVD++ (a SVD model with time features). Due to the high similarity between SVD and SVD++, we proposed the use of NCF instead, while the additional time effect bias in SVD++ is left out as a to-do.

1.3. Notations

In this report, we denote variables about the dataset as follows:

- m: number of users

- n: number of movies

- N: number of ratings

- r_{ij}: rating of user i on movie j

2. Data preprocessing

Due to performance issues with CSV parsing, we wrote a separate program 'preprocess-dataset', which reformats the input files into a numpy saved array which contains the following columns derived from the original dataset:

- User ID

- Movie ID

- Rating value

- Year of rating

- Day of year of rating

- Day of week of rating

- Whether the rating is probed data

During runtime, the data are represented in compressed sparse row (CSR) format for a sparse $m \times n$ matrix.

3. KNN

KNN is an example of neighbourhood models. It seeks to provide recommendations based on the behaviour of similar users. In KNN model, we assume that similar users will give close ratings on similar movies. To predict the rating $\hat{r}_{ij}$ of user i on movie j ($1 \le i \le m, 2 \le j \le n$), the k most similar users ($n_1, \ldots, n_k$) to user i are computed based on similar choices of ratings, and $\hat{r}_{ij}$ is estimated based on the known values $r_{n_1 j}, \ldots, r_{n_k j}$.

3.1. Algorithm

The training algorithm involves the following steps:

The algorithm first finds q, the set of q users with the highest number of ratings. The distance $d(i, j)$ for $i \in [1, m], j \in q \setminus \{i\}$ is then computed, and the neighbourhood $K_i \subset q$ of each user i is evaluated as the k users with the lowest distance. The prediction of the rating r_{ij} is computed by aggregating the neighbourhood ratings with $A(\{r_{xj} : x \in K_i\})$. The memory complexity of these operations (except d and A) is $O(qn)$.

In this KNN implementation, there are four hyperparameters to be considered, namely neighbourhood size k, neighbourhood candidate size q, distance function d and aggregation function A.

3.1.1 Neighbourhood size k

k refers to the number of nearest neighbours to be included in recommendation system.

A large value of k would reduce accuracy as users with lower similarity are selected. On the other hand, a small value of k would be biased over the choice of the most similar user. We have tested with different values of k with different values of the other hyperparameters.

As a baseline model, we also set $k = m$, i.e. to evaluate the RMSE by taking the plain arithmetic mean of all users.

3.1.2 Neighbourhood candidate size q

q means the number of neighbour candidate the recommendation system considered. We tried different values of q, which led to different sets of k-nearest neighbours being selected each time.

3.1.3 Distance function d

The distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ observes the deviation in interests between two different users. $d(i, j) = 0$ iff $i = j$, and $d(i, j) = d(j, i) \forall i = j$. Although $d(i, j) < d(i, k)$ implies that j is more similar to i than k is, it is not necessary that d follows triangular inequality.

Cosine The cosine distance is defined as

$$d(\mathbf{x}, \mathbf{y}) = 2 \frac{(\mathbf{x} - \mathbf{b}) \cdot (\mathbf{y} - \mathbf{b})}{\|\mathbf{x} - \mathbf{b}\| \|\mathbf{y} - \mathbf{b}\|} - 1$$

where $\mathbf{b} = \frac{1}{n} \sum_{i=1}^{m} \mathbf{r}_{i\cdot}$, i.e. the mean rating of each movie. This metric emphasizes the difference between users who gave ratings better or worse than the average, e.g. the difference between 3 and 5 is more important than the difference between 1 and 3 when the average rating is 4. It can be thought of a metric to determine how "abnormal" a user is. The multiplier is to correct the distance range to $[0, 1]$.

p-norm The p-norm distance is defined as the p-norm of two vectors normalized by the maximum possible distance, i.e.

$$d(\mathbf{x}, \mathbf{y}) = \frac{(\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}}{4n^{1/p}}$$

.

3.1.4 Aggregation function A

We computed both baseline and average aggregation functions.

3.2. Findings

3.2.1 Baseline model result

The RMSE value attained in baseline model is 1.0528.

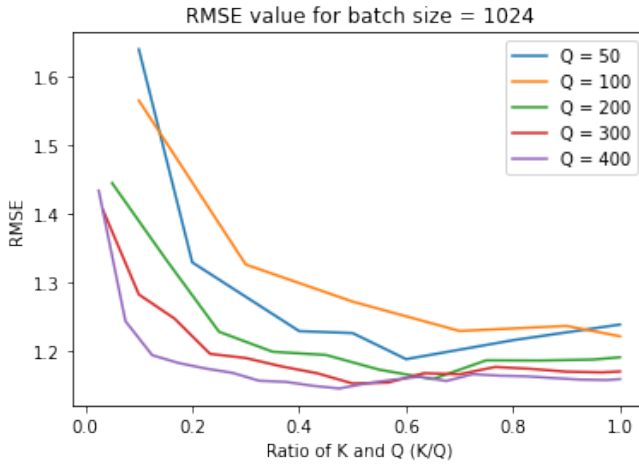#### 3.2.2 Cosine distance model result

Figure 4. RMSE for different combination of q and k with cosine distance function

We can see the RMSE values decreased with the increasing q value in general. The curve with $q = 400$ has the smallest values for nearly all ratio of k and q. The curve with $q = 50$ however has larger values for most of the ratio

values than the curve with $q = 100$. All the curves have non-smooth logarithmic curves with fluctuations between ratio = 0.1 to ratio = around 0.7. The lowest RMSE value attained with the cosine distance function is 1.1455 with ratio = 0.475 ($q = 400$ and $k = 190$). For the other values of q, when $q = 300$, the lowest value found in ratio = 0.5 ($k = 150$). When $q = 200$, the lowest value belonged to ratio = 0.65 ($k = 130$). When $q = 100$, the lowest value attained at ratio = 1 ($k = 100$). When $q = 50$, the lowest belonged to ratio = 0.6 ($k = 30$). To conclude, the ratios for the lowest value of RMSE for each curves varied.

4. SVD

4.1. Background

4.1.1 Learning rate α

4.1.2 Regularization λ

4.1.3 Rank of factorization k

4.1.4 Number of epochs ξ

4.1.5 Batch size β

4.2. Findings

5. NCF

Among all collaborative filtering techniques, matrix factorization (MF) become a standard approach to the model-based recommendation. Many research teams have put their efforts on enhancing it to break the accuracy record.

However, it has several limitations such as the cold-start problem – how the system provide recommendations to new users, how to recommend when new item is added without feature vector or embedding [4]. Also, the use of dot product for recommending popular items will lead to the tendency of recommending popular items to those users without specific user interests. With the use of inner product of User and item feature embeddings, MF limits the capture of the complex relations of the users and items interactions [2].

In this assignment, we aim to implement the new approach, Neural Collaborative Filtering (NCF), proposed by He team in 2017 [2]. They suggested that their design of deep neural network can overcome the limitation of MF.

5.1. Model structure

This is the model proposed by He team in 2017 [2]. Their proposed model uses a combination of matrix factorization and multi-layer perceptron techniques, and the hidden layers from the two techniques are combined by a fully-connected layer. However, their paper mainly focuses on the prediction of *implicit user feedback*, that is, they are

predicting the values of y_{ui} where $y_{ui} = 1$ when the obervation (user u, item i) is observed, and 0 when the observation (user u, item i) is not observed. Thus, at the final stage of their proposed model, the output of the fully-connected layer is passed into a sigmoid function, such that the model only outputs values within $[0, 1]$.

The problem of predicting implicit feedback is different from the problem we wish to solve in the Netflix dataset, because in the Netflix problem we wish to predict actual ratings from $[1, 5]$. Nevertheless, we still think that He's paper provides a good starting point - We modify He's model by removing the final sigmoid function and changing the loss functions used directly, such that the neural network solves a regression problem instead of the original binary classification problem.

The model proposed by He is split into 3 parts - Generalized Matrix Factorization, Multi-Layer Perceptron, and Neural matrix factorization. In fact, the third part is a combination of the previous two parts. For this part of our project, we will follow the same workflow as He and implement the 3 parts separately.
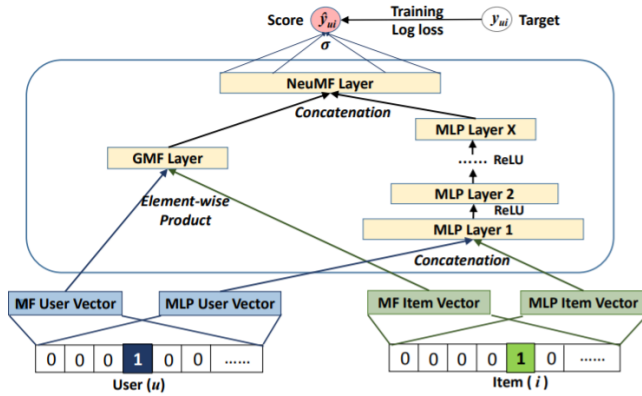

Figure 5. Neural Collaborative Filtering Model Design [2]

5.1.1 Model - Input Layer

The input of the model are the user ID and item ID transformed to a binarized vector with one-hot encoding.

5.1.2 Model - Embedding Layer

For each user and each movie, we train two sets of embedding latent vectors. This is because one set of latent vectors will be passed into the itemwise multiplication part of the model (in Hes paper, it is called Generalized Matrix Factorization (GMF)), and the other set of latent vectors will be passed into the neural network part of the model (in Hes paper, it is called Multi-Layer Perceptron (MLP)).

5.1.3 Model - Generalized Matrix Factorization (GMF)

In this part of the model, the user latent vector and item latent vector are combined via itemwise multiplication, similar to that in the SVD matrix factorization model. The output of this part is another vector with the same size at the latent vectors and is passed on to the final output layer of the model.

5.1.4 Model - Multi-Layer Perceptron (MLP)

In this part of the model, the user latent vector and item latent vector are first concatenated. Afterwards, the whole concatenated vector is passed through several layers of fully connected hidden linear layers, using the ReLU activation function. the output of this part is a vector dependng on the output size of the last hidden layer and is passed on to the final output layer of the model.

5.1.5 Model - Output Layer

The final output layer takes in the vectors produced by the GMF and MLP parts of the model and concatenates them. The output of this part, which is the final output of the whole model, is a weighted sum of the vector elements, where the weights are model parameters to be learned. The final output layer produces a predicted score $\hat{y_{ui}}$.

5.1.6 Model - Formula

The formula of the GMF is as follow:

$$\hat{y_{ij}} = f(P^T v_i Q^T v_j | P, Q, \Theta_f)$$

The variables with m users and n items are denoting as:

- $P \in \mathbb{R}^{(m \times K)}$ latent factor matrix for users from embedding layer.

- $Q \in \mathbb{R}^{(n \times K)}$ latent factor matrix for items from embedding layer.

- f interaction function.

- Θ_f model parameters of the interaction function f

For MLP f, it can be formulated as:

$$f(P^T v_i, Q^T v_j) = \phi_{output}(\phi_X(...\phi_2(\phi_1(P^T v_i, Q^T v_j))...))$$

The mapping functions are as follow:

- ϕ_{output}: the mapping function for output layer

- ϕ_x the activation function for the xth hidden neural FC layer. x is the number of the layers

5.2. Technical Details

The movies are loaded into a list with columns of Movie ID, User ID and Rating. Then, it will be fed into the test loader with batch size.

We select Adadelta as our optimizer. To have a better performance on the converge of loss function,we implement a scheduler with different combinations of hyperparameters of scheduler.

5.3. Findings

6. Conclusion

References

[1] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at AT&T Research*, 1(1), 2008. 2

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017. 4, 5

[3] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008. 2

[4] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval*, SIGIR '11, pages 315–324. ACM, 2011. 4