Recommendation System For Restaurants Using Various Neural Networks

Lam Nguyen
tln3@sfu.ca

Phu Khang Bui
pkbui@sfu.ca

Phan Bui
dbui@sfu.ca

Abstract

The idea of recommendation system using collaborative filtering has various methods of implementation, and are still continuously developed. In this work, we implemented two different algorithms for collaborative filtering - matrix factorization and neural network, with deep-layer neural network utilizing two techniques of batch normalization and early stopping in separate instances. We examine the effectiveness of these implementations by comparing their results in performing on Yelp dataset. Result shows that while all methods generate good results in reducing loss, layered neural network products a better result by small margin compared to other models.

1 Introduction

The concept of recommendation system appears almost in every aspects our modern life. Netflix recommendation for a movie night, Spotify suggesting new music based on your current playlist, and many other examples are all instances of recommendation systems, and are utilized regularly by users of these applications. Services continously try to improve their ability to predict the preferences of their users in order to provide a better experience, and this has created an interest in developing more efffective recommendation systems. Recommendation system can be largely divided into two types: content-based and collaborative filtering[1]. We focus on collaborative filtering in this work, as this is more often considered to be the more effective methods, and particularly memory-based collaborative filtering. While matrix factorization technique[2], a very common effective approach to collaborative filtering problem, has already been applied widely in many recommender systems, there is still not many researches for using neural network to tackle this problem. With resources and researches available online, we decide that this would be an interesting topic to implement and examine the potential of deep neural network to solve collaborative filtering problem as well as compare performance of different models to conclude which one produces a better result.

2 Approach

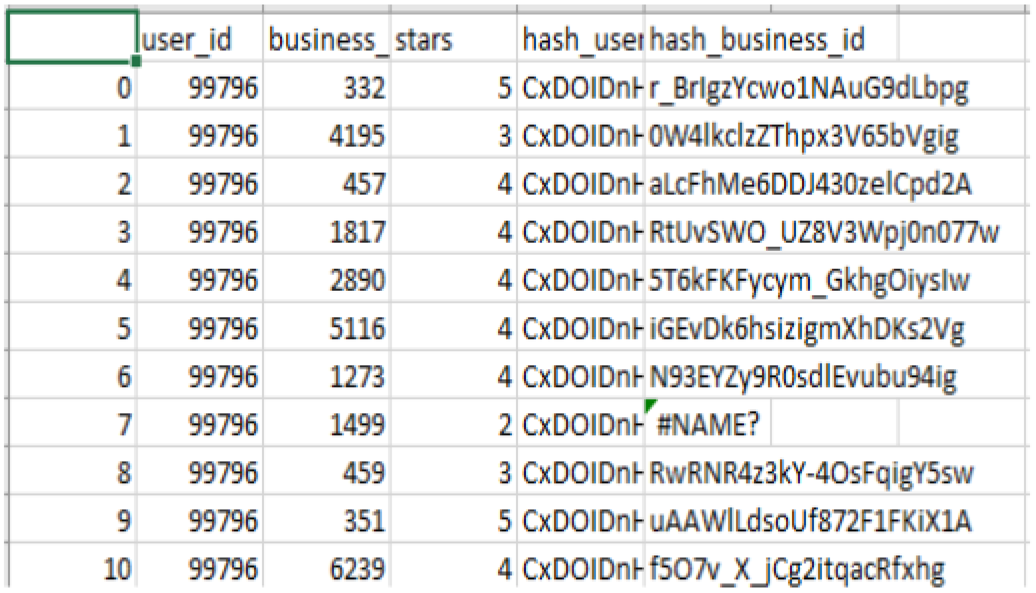
2.1 Dataset and features

The dataset we used in this project comes from a public dataset provided by Yelp, which is a web application allowing users to find and review restaurants. The data provides information about relationships between users (such as which users are friends with each other), which provides benefits in identifying ratings that are more relevant to an user (such as their friends' preference are more likely to be similar to their preference) and therefore constructing a more relevant matrix.

The dataset we used in this project comes from a public dataset provided by Yelp, which is a web application allowing users to find and review restaurants. The Yelp public dataset is composed of almost 6,000,000 reviews with 1,518,069 users and 188,593 restaurants. All the records of review are stored in a JSON file in which each object contains information about a review such as review_ID, user_ID, restaurant_ID, rating stars, etc. When we constructed the dataset as a huge matrix with columns as restaurants and rows as users, we found that the dataset is significantly sparse, which means that the amount of information we have is extremely smaller than the amount of unknown features.

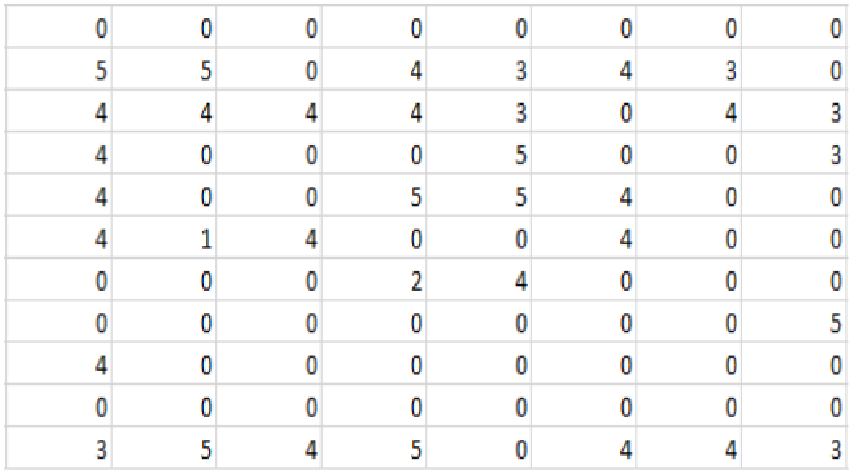
2.2 Data processing

The data was first stored as a JSON file so we converted it into a CSV file which allowed us to process the data more conveniently. We observed that the dataset has a huge number of records, however most of them have no relevant connection with each other. For example, users' location can be from anywhere and may not be relevant to the user we are recommending. In that case, it would make no sense if we tried to build a model, for example, using information of an user living in Vancouver to recommend for another one living in New York. We have to take geolocation relevance into account. In order to solve this problem, we decided to choose the users who have friends in their friend-lists and have rated at least 5 restaurants. The reason for this is because the users who are friends of each other will be more likely to live in a same region and have the same preference for food. By adding these constraints, we managed to have a less sparse, more relevant and meaningful information among users and restaurants. We also utilized fast and high-reliability SQLite database as our choice to store records of reviews, which help speeding up querying and computation time. After preprocessing, we achieved two data files as shown below:



	user_id	business_	stars	hash_user	hash_business_id
0	99796	332	5	CxDOIDnH	r_BrlgzYcwo1NAuG9dLbpg
1	99796	4195	3	CxDOIDnH	0W4lkclzZThpx3V65bVgig
2	99796	457	4	CxDOIDnH	aLcFhMe6DDJ430zelCpd2A
3	99796	1817	4	CxDOIDnH	RtUvSWO_UZ8V3Wpj0n077w
4	99796	2890	4	CxDOIDnH	5T6kFKFycym_GkhgOiysIw
5	99796	5116	4	CxDOIDnH	iGEvDk6hsizigmXhDKs2Vg
6	99796	1273	4	CxDOIDnH	N93EYZy9R0sdlEvubu94ig
7	99796	1499	2	CxDOIDnH	#NAME?
8	99796	459	3	CxDOIDnH	RwRNR4z3kY-4OsFqigY5sw
9	99796	351	5	CxDOIDnH	uAAWlLdsoUf872F1FKiX1A
10	99796	6239	4	CxDOIDnH	f5O7v_X_jCg2itqacRfxhg

Figure 1: User_id - restaurant_id - rating CSV file



0	0	0	0	0	0	0	0
5	5	0	4	3	4	3	0
4	4	4	4	3	0	4	3
4	0	0	0	5	0	0	3
4	0	0	5	5	4	0	0
4	1	4	0	0	4	0	0
0	0	0	2	4	0	0	0
0	0	0	0	0	0	0	5
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
3	5	4	5	0	4	4	3

Figure 2: User-restaurant rating matrix

3 Building and training model

3.1 Normalizing input

For building our model, first we started with constructing these two normalized matrices based on user-restaurant matrix by using this formula[3]:

$$User_i = User_i \circ \frac{M}{\|User_i\|_0}$$

$$Res_j = Resj \circ \frac{N}{\|Res_j\|_0}$$

$User_i$ is the ith row of the matrix
$Res_j$is the jth column of the matrix
M is number of restaurants (columns)
N is number of users (rows)
$\|User_i\|_0$ is the number of non-zero entries of the vector $User_i$
$\|Res_j\|_0$ is the number of non-zero entries of the vector Res_j

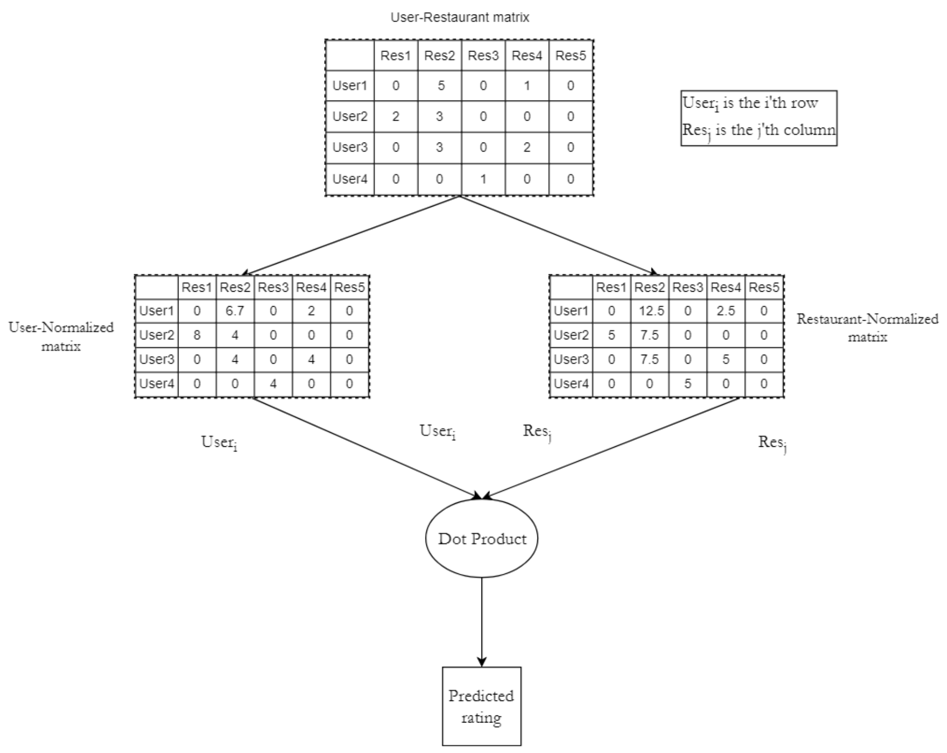
The reason why we used normalized input is because we have a very small number of collected ratings compared with missing ratings, which basically means there are significantly more zeros than non-zeros entries in matrix. The problem of missing ratings may result in inefficiently-trained mode due to highly sparse input. There is an approach to mitigate this problem by treating missing ratings as dropout[4] units with probability computed by this formula:

$$probability = \frac{r}{n}$$

where r is the number of missing ratings and n is number of total items.

3.2 Matrix Factorization Collaborative Filtering Model

After normalizing the input inputs, we started to build the models for matrix factorization technique and deep neural network. Matrix factorization has recently been a very efficient method used in recommendation system by exploiting the latent factors of users and items[5]. The key idea behind matrix factorization lies on the assumption that there exists implicit features which denote users' preference for particular items, that is not captured fully by the rating scores. In our matrix factorization model, the normalized inputs served as latent factors and outputs were computed by dot product of two latent-factor vectors $User_i$ and Res_j. A visualization of model is shown below:



	Res1	Res2	Res3	Res4	Res5
User1	0	5	0	1	0
User2	2	3	0	0	0
User3	0	3	0	2	0
User4	0	0	1	0	0

	Res1	Res2	Res3	Res4	Res5
User1	0	6.7	0	2	0
User2	8	4	0	0	0
User3	0	4	0	4	0
User4	0	0	4	0	0

	Res1	Res2	Res3	Res4	Res5
User1	0	12.5	0	2.5	0
User2	5	7.5	0	0	0
User3	0	7.5	0	5	0
User4	0	0	5	0	0

Figure 3: Matrix factorization model

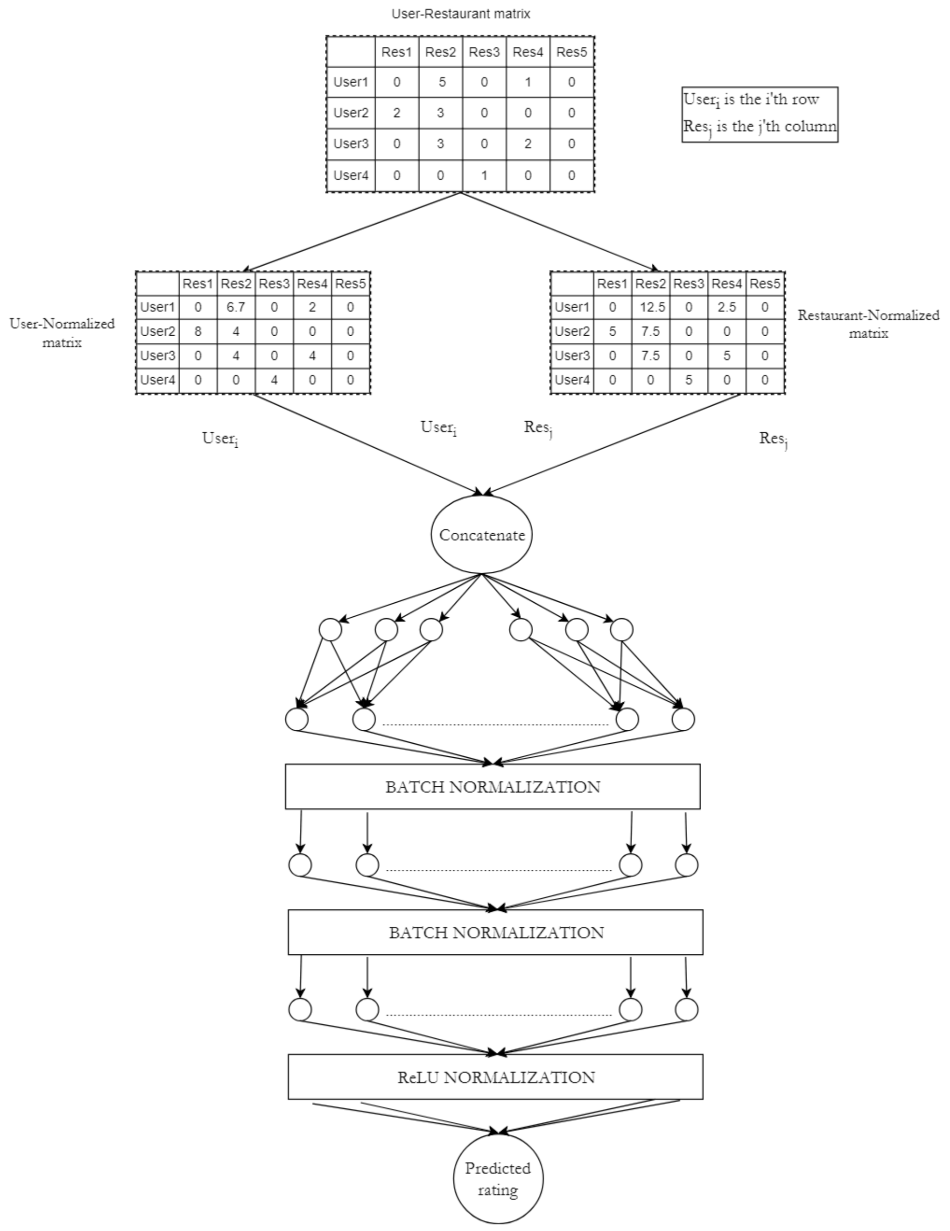
The model was trained by minimizing loss function which was constructed based on the difference between original ratings and predicted ratings.

3.3 Neural Network Collaborative Filtering Model

We also investigated on the potential of utilizing neural network in collaborative filtering technique. After trying different implementations, we finally achieved an efficient model using 5 layers. The input layer for neural network model takes the concatenation of each column and row from normalized user-restaurant matrix. This is different from matrix factorization model, for which we used dot product computation, because we want our neural model to learn how changing a latent feature would affect rating prediction. We built three hidden layers with 64, 32 and 16 units, respectively and the output layer was one unit. In all hidden layers, we applied dropout[6] and batch normalization[7] with a view to preventing model from overfitting. In this model, we used only one activation function which is ReLU for the output layer. The loss function we chose to minimize is Root Mean Square Error(RMSE) [8] which is computed as below:

$$RMSE = \sqrt{\frac{1}{|S_{test}|} \sum_{(u,i) \in S_{test}} (r_{u,i} - r'_{u,i})^2}$$

Where S_{test} is the testing set, $(u,i) \in S_{test}$ means that rating of user u on restaurant i is in the testing set, $r'_{u,i}$ is the predicted rating and $r_{u,i}$ is the actual rating.



User-Restaurant matrix

	Res1	Res2	Res3	Res4	Res5
User1	0	5	0	1	0
User2	2	3	0	0	0
User3	0	3	0	2	0
User4	0	0	1	0	0

User-Normalized matrix

	Res1	Res2	Res3	Res4	Res5
User1	0	6.7	0	2	0
User2	8	4	0	0	0
User3	0	4	0	4	0
User4	0	0	4	0	0

Restaurant-Normalized matrix

	Res1	Res2	Res3	Res4	Res5
User1	0	12.5	0	2.5	0
User2	5	7.5	0	0	0
User3	0	7.5	0	5	0
User4	0	0	5	0	0

Figure 4: Neural network using Batch Normalization model

4 Result and Analysis

After training the two models with different implementations, we achieved some interesting results shown below:
-Using matrix factorization model, we obtained a very good result with minimum of 0.95 and 1.02 loss for training set and test set, respectively. Figure 5 shows how our model improved its accuracy and decreased loss over epochs. (figure 5)
-For the neural network model, before training started we only expected it to be as good as matrix factorization. But more than our expectation, the neural network model gave us impressive results and even outperformed matrix factorization.
-When we trained model with Batch Normalization and ReLU activation, we already achieved a good result with approximately 0.65 and 0.68 loss on training data and testing

data, respectively (figure 6). However, we realized that we could reach a better performance with significantly faster training process by using Early Stopping technique, which gave us 0.64 loss on testing data. (figure 7)

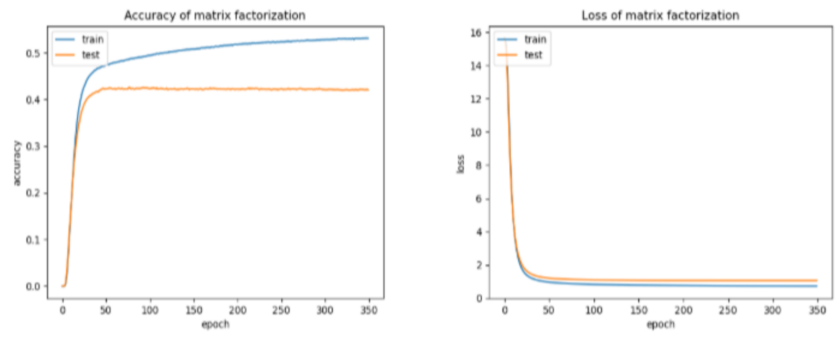


Figure 5: Matrix factorization model result

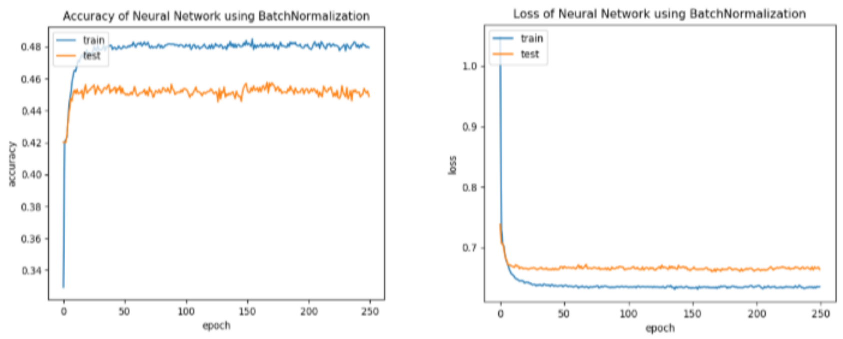


Figure 6: Neural network model with Batch Normalization result

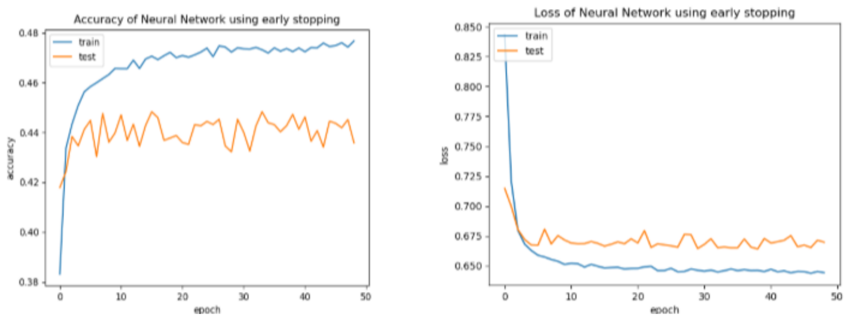


Figure 7: Neural network model with Early Stopping result

- During the training, we also learned that training model with too complicated layers (e.g using too many inputs) or using inappropriate activation functions could lead to overfitting problem. We solved overfitting and boosted training process by reducing the number of inputs of each layer, using Drop-out, Batch Normalization, Early Stopping and ReLU activation function.

5 Conclusion and Future Work

- We learned that applying Neural Network in Collaborative Filtering problem potentially produces good performance for a recommendation system and slightly outperformed Matrix Factorization technique. The Neural Network model performed quite well on small dataset with roughly 120,000 records. For our next step, we want to train the model on a larger and sparser dataset to see if the model can still perform well.
- In addition, the training process took us a lot of time since we were using a very large User-Restaurant matrix as the input for neural network. In the future, we want to use latent-feature vectors of User and Restaurant as the embedding layers for inputs, which is expected to speed up the training stage and might produce a better result. Inspired by work of Xiangnan, H. et al [9], we also plan to combine the result of matrix factorization with the inputs of neural network layers, which gives more information about user-item relationship for neural network model to learn.
- For the third improvement, we would want to implement live demo for our model's recommendation, to showcase how well the model performs with real-time data, if possible.

6 Contribution

- Lam Nguyen: Researching for project topic, filter the original dataset for relevant information, and implement the majority of our models training, writing reports.
- Khang Bui: Researching for project topic and looking up datasets, partly works on implementation of models training, writing reports.
- Phan Bui: Partly works on models training implementation, designing posters, writing reports.

7 References

[1] Ekstrand, M.D, Riedl, J.T & Konstan, J.A (2011) Collaborative filtering recommender systems, Foundations and Trends in Human–Computer Interaction, vol. 4, no. 2, pp 81-173.

[2] Yehuda, K., Robert, B. & Chris, V. (2009) Matrix Factorization Techniques For Recommender Systems, vol. 42, pp. 30-37, IEEE.

[3] [8] Lee, H., & Lee, J. (2018). Scalable deep learning-based recommendation systems. ICT Express. Retrieved December 14, 2018, from https://www.sciencedirect.com/science/article/pii/S2405959518302029#b14.

[4] [6] Nitish, S., Geoffrey, H., Alex, K., Ilya, S. & Ruslan, S. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overffiting, vol. 15, Journal of Machine Learning Research 15.

[5] [9] Xiangnan, H., Lizi, L., Hanwang, Z., Liqiang, N., Xia, H. & Tat-Seng, C. (2017) Neural Collaborative Filtering, International World Wide Web Conference Committee.

[7] Sergey, L. & Christian, S. (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, vol. 37, pp. 448-456, International Conference on Machine Learning.