UNIVERSITY OF CALIFORNIA

Los Angeles

Application of Neural Network Based Recommendation System

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Chufeng Hu

2017

ABSTRACT OF THE THESIS

Application of Neural Network Based
Recommendation System

by

Chufeng Hu
Master of Science in Statistics
University of California, Los Angeles, 2017
Professor Yingnian Wu, Chair

In the recommendation system, data comes in the form of a vector or matrix. Matrix factorization techniques attempt to recover missing or corrupted entries by assuming that the data matrix is the linear product of two independent matrices. The idea is to replace those matrices by two arbitrary functions that we learn from the data at the same time as we learn the latent feature vectors. The resulting approach is called Bi-generator neural network. In this paper, I made several attempts to introduce this technique to the MovieLens datasets. The result shows that Bi-generator can be very close to some recent proposals that also take advantage of neural network. Due to the limit of computational power, I mainly focus on 2-layer neural networks in this paper. Given the vast range of neural networks architectures, it seems likely my experiments have not identified the limitation of Bi-generator model.

The thesis of Chufeng Hu is approved.

Jingyi Li

Qing Zhou

Yingnian Wu, Committee Chair

University of California, Los Angeles

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Ying Nian Wu for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Tian Han for introducing me to the topic as well for the support on the way. Also, I would like to thank my loved ones, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

CHAPTER 1

Introduction to Recommendation System

Recommendation systems are applications that involve predicting the rating or preference that a user would give to an item. The most known applications are recommendation algorithms for Amazons online shopping, Netflixs movie, Twitters news and Yelps restaurants. The long-tail phenomenon makes recommendation system necessary [1]. We are living in the age of information and entering the age of recommendation. Physical stores have limited space, thus can show the costumer only a small fraction of all the choices. On the other hand, online stores can make anything that exists available to the customer. The demand of customers can be described in a long-tail distribution, and it is suggested in Figure-1. This phenomenon force on-line institutions to recommend items to individual users, since we dont expect users to already know all the items they might like. Furthermore, e-commerce companies are not the only ones that use recommendation engines to persuade customers to buy additional products [1]. Recommendation systems have been widely used in other industries, from recommending music and events to dating profiles.

The data for recommendation systems can be presented in a rating matrix, its rows and columns are users and items. Users have ratings (preferences) for certain items. The recommendation system can extract users preference information from data matrix. The rating of a user-item pair describes the degree of preference of that user for that item. Values of rating can come from an ordered set from 1-5, or just Boolean variable with 0 (dislike) and 1 (like). The data is usually a sparse matrix, since most users have ratings on only a subset of items from the pool of all available items, the other entries without a rating are unknown. The goal of recommendation algorithms can use users preference information to predict what rating a user would give to a previously unrated item, thus the system can

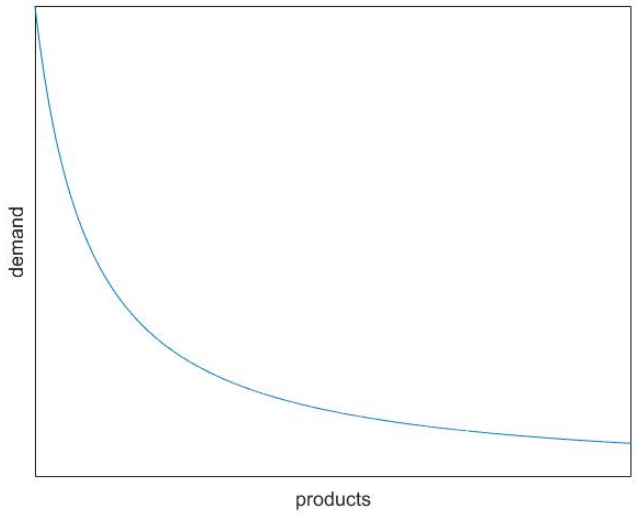

Figure 1.1: The long tail: physical store can only provide what is popular, while online institutions can make everything available.

recommend users with item they might like.

Most recommendation systems typically produce a list of recommendations in one of two ways collaborative and content-based filtering. Some systems combine both of these approaches (Hybrid recommender system), some systems use knowledge-based recommendation.

1.1 Collaborative Filtering

Collaborative filtering is based on the idea of word-of-mouth promotion. That is the opinion of family members and friends plays a major role in personal decision making [2]. In applications, family members and friends are replaced by nearest neighbors. Thus the model can be constructed from the behavior of other users who have similar traits, and the model can be either user-based or item-based [2].

User-based collaborative filtering identifies the k-nearest neighbors of the target user and calculates a prediction of the target users rating on certain items (usually a weighted average). In contrast to user-based collaborative filtering, item-based collaborative filtering

Rating Data					
User/Item	I_1	I_2	I_3	I_4	I_5
U_1	1	?	?	3	?
U_2	4	3	?	1	?
U_3	?	4	5	?	2
U_4	2	3	1	5	?
U_5	?	2	3	?	1

Table 1.1: Table 1: A data matrix representing ratings of items on a 1-5 scale. Every row contains each user's ratings on different items. Every column contains each item's received ratings from different users. Question marks represent the situation where the user has not rated the item.

searches for k nearest neighbors of the target item.

User-based collaborative filtering suffers from scalability problems as the user base grows. Searching for the neighbors of a user can become very expensive when system has a very huge user database, and adding one user may change many neighbors of other users. On the other hand, item-based collaborating filtering is more stable [17], one user adding or changing ratings is unlikely to significantly change the similarity between two items. Therefore, item-based collaborative filtering is now more preferable than user-based collaborative filtering.

1.2 Content-based Filtering

Content-based systems are based on the assumption of monotonic personal interests [18]. In a content-based system, we first construct for each item a profile, which is a record or collection of tags representing important characteristics of that item. Then content-based filtering recommendation systems calculate a set of items that are most similar to items already consumed (or highly rated) by certain customer. For example, if a customer likes Star Wars, then he/she would most likely to like fiction movies such as Guardians of the Galaxy.

Content-based filtering is also widely used in many recommender systems. Pandora Radio is a popular example of a content-based recommender system for music [20]. Therere also other recommendation system for movies, videos, news etc. Public health professionals have been studying recommender system to personalize health education and preventative strategies.

1.3 Hybrid Recommender Systems

Hybrid recommendation systems are the combination of collaborative filtering and content-based filtering. It would be more effective than a single filtering methods in some cases. Hybrid approaches can be implemented by making collaborative-based and content based filtering separately and then combining them. It can also be used to address collaborative filtering that starts with space data by enabling the results to be weighted initially toward content-based filtering, and shifting the weight toward collaborative filtering as the available user dataset matures. Some studies empirically compare the performance of the hybrid with pure collaborative and content-based system and demonstrate that the hybrid can provide more accurate recommendations than pure approaches [3].

CHAPTER 2

Algorithms for movie recommendation system

What movie should we watch this weekend? What movie should a seller recommend to a certain costumer? A recommendation system engine will help people find appropriate video that can fit their taste, and save time when searching on the endless list movie list on Netflix, Rotten Tomatoes and Hulu. Most movie recommendation systems are based on one of two strategies: content filtering and collaborative filtering, some recommendation system use the combination of them (Hybrid). The major appeal of collaborative filtering is that it is domain free, and its generally more accurate than content-based techniques. [10]. There are two primary areas of collaborative filtering, they are neighborhood methods and latent factor models.

2.1 Neighborhood methods

The neighborhood methods focus on the similarities between movies or users. The item based approach will recommend customer with movies which are the neighbors or the preferred movies or current customer. The user based approach will recommend movies to the user that have been highly rated by the users neighbors. The neighbors of movie and user are defined as the most K similar movies to the current movie, and the most K similar users to the current user (where K is a predefined parameter). Similarity between users can be calculated by Pearson correlation:

$$s(u,v) = \frac{\sum_{i \in I_u \cup I_v}(r_{u,v} - \hat{r}_u)(r_{u,v} - \hat{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{u,v} - \hat{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{u,v} - \hat{r}_u)^2}} \tag{2.1}$$

Pearson correlation suffers from computing high similarity between users with few ratings

in common. This can be alleviated by setting a threshold on the number of co-rated items necessary for full agreement and scaling the similarity when the number of co-rated items falls below this threshold [11].

2.2 Matrix factorization

Latent factor models try to find factors that inferred from the rating patterns of movies. These latent factors can be used to characterize both movies and users. For movies the factors might measure obvious dimensions such as comedy versus drama, amount of actions, or orientation to children [10].

Matrix factorization is one of the most important approaches to discover latent factors. This method is popular for combining good scalability with predictive accuracy. Furthermore, it offers much flexibility for modeling various applications.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, and the interactions (ratings) of users and items is an inner production of users and items [10]. For example, in a movie recommendation system, the matrix factorization would map a user i into latent space with five dimensions: v_i, where each dimension can be interpreted as the level of preference to action, adventure, comedy, horror, science fiction, And a movie j can also be mapped to that space: v_j, where each dimension measures the extent to which the movie possesses those 5 factors. And the rating $r_{i,j}$ the user i would give this movie j is the inner production of u_i and v_j.

$$\hat{r}_{i,j} = u_i^\top v_j \tag{2.2}$$

The major challenge is computing the U and V, given the rating matrix R. To learn the latent factors, the system minimizes the distance between actual ratings and estimated ratings plus the regular term:

$$\min_{U,V} \sum_{i,j} (r_{i,j} - u_i^\top v_j) + \lambda(\|u_i\|^2 + \|v_j\|^2) \tag{2.3}$$

The regular term prevents overfitting by regularizing the learned parameters. Both stochastic gradient descent (SGD) and alternating least squares (ALS) can minimize the above equation. Another challenge is that much of the observed variation in rating values is due to effects associated with either users or item. For example, some user intends to give movies higher ratings than others, and some movie receive lower ratings than others. This is known as biases or intercepts, which is independent of any latent factors. Therefore, the formula for prediction can be rewrite as

$$\hat{r}_{i,j} = b_i + b_j + u_i^\top v_j \tag{2.4}$$

In some situations, biases tend to capture much of the observed signal, their accurate modeling is vital. Hence some systems use more elaborate bias models [10].

CHAPTER 3

Neural Network based recommendation systems

Recommendation engines have different algorithms even they are based on the same approaches. The design of these recommendation systems depends on the markets and data theyre involving with. And many researches focus on the application of neural networks in recommendation system. In this chapter I will first make brief introduction to neural networks, then I will discuss two applications of neural networks in recommendation system, then I will introduce the Bigenerator Net that I applied to the Movie rating data.

3.1 Introduction to neural networks

Neuron is the basic unit of the neural networks. In the computational model of a neural network, the signals (input data vector X) that travel along the axons interact multiplicatively with the dendrites of the other neuron based on the different structure of the neural units (weights W). Neurons are learnable and control the strength of the influence and direction of signals in the system. In a basic model, the data comes to those neurons where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending output data as predicted classes or the input of next layer of neurons. The firing rate of a neuron can be an activation function f, which represents the frequency of the signal fire from the neuron. Some activation functions are: Tanh, ReLU and Maxout.

3.1.1 Activation functions

The sigmoid non-linearity has the mathematical form $\sigma = \frac{1}{1+e^{-x}}$. The function takes a real-valued input and squashes it to range between 0 and 1. Sigmoid function has a nice

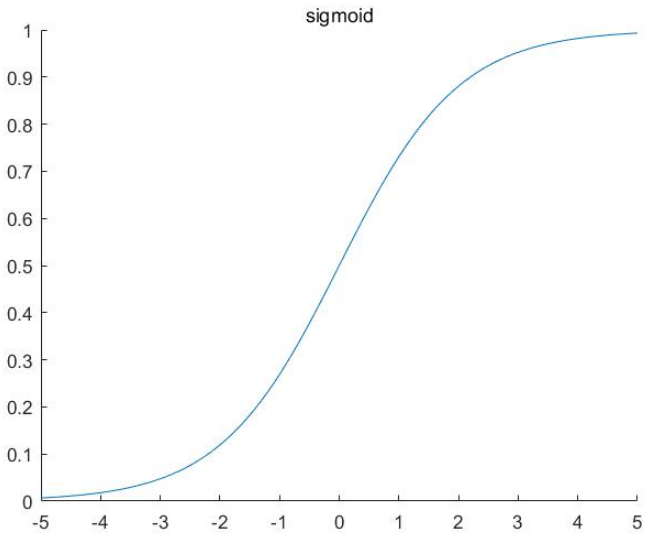

Figure 3.1: Sigmoid function, it's not zero centered. The gradients of sigmoid function saturate near 0 and 1.

interpretation as the firing probability: from not firing at all (0) to fully saturated firing (1). However, in practice, the sigmoid has two major drawbacks: Sigmoid function can saturate gradients, when the firing rate is either close to 0 or 1, the gradient at these regions is almost zero; Sigmoid outputs are not zero-centered. Because of these two drawbacks, sigmoid has recently been less used than other functions

Tanh is simply a scaled sigmoid neuron and is preferred to the sigmoid function. The function $tanh = 2\sigma(2x) - 1$ can convert a real-valued number to the range $[-1, 1]$. Like the sigmoid neuron, it will also saturate gradient, but its zero centered.

The rectified Linear Unit computes the function $f(x) = \max(0, x)$. Its simply thresholded at zero. The advantages of ReLU are: ReLU can greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid/tanh functions [4], this is due to the linearity of ReLU. Also compared with the complex operations in Sigmoid/tanh, ReLU can be easily interpreted and implemented in practice. However, ReLU units can be fragile during training and can die. For example, the forward data and backward gradients flowing through the ReLU unit may update the weights in such a way that output of ReLU unit will always be zero, in that case the neuron dies.

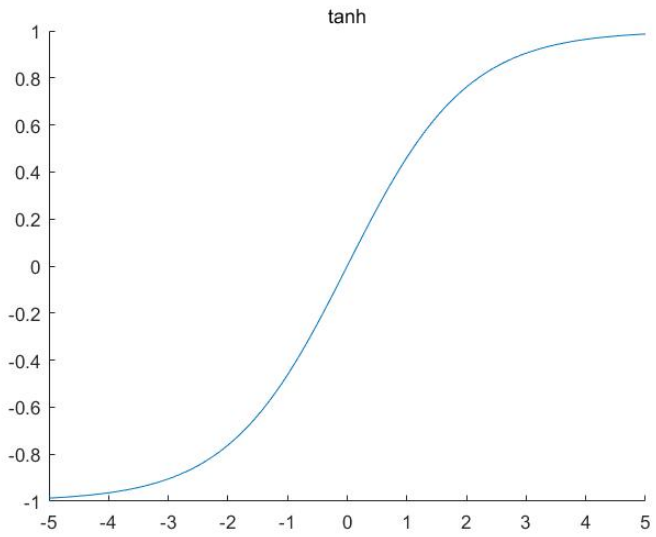

Figure 3.2: Tanh function, it's zero centered, but gradients still saturate near -1 and 1

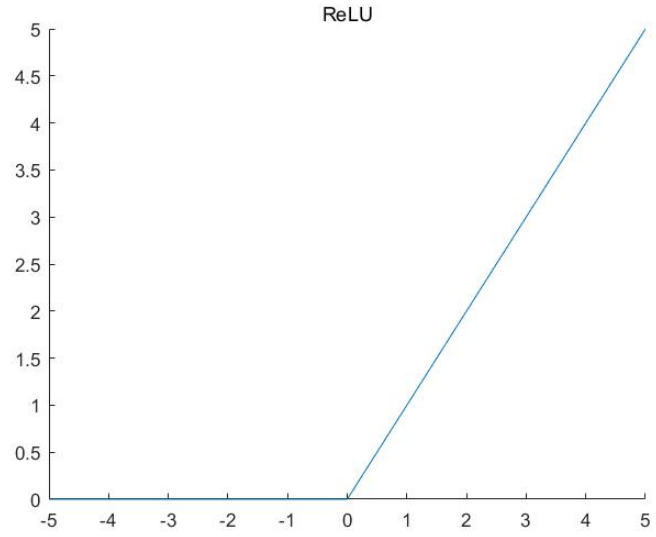

Figure 3.3: ReLU function is a simple threshold at zero

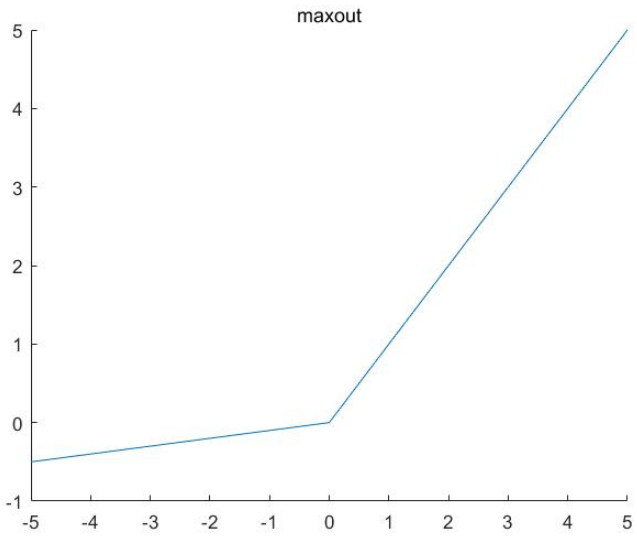

Figure 3.4: Maxout function generalizes the ReLU and its leaky version

Maxout computes the function $\max(w_1^\top x + b_1, w_2^\top x + b_2)$, it generalize the ReLU and its leaky version. ReLU can be seen as a special version of a ReLU unit (where $w_1^\top b_1 = 0$). Therefor Maxout neuron enjoys all the benefits of a ReLU unit (linearity and no saturation) while does not have its drawbacks (fragile). But using Maxout activation function will lead to a high total number of parameters.

3.2 Neural networks architectures

Neural Networks are constructed as collections of neurons that are connected in a graph. Neural Networks are often built into distinct layers of neurons, and neurons in the same layer are not allowed to have connections. For regular neural networks, the most common layer type is the fully-connected network, where neurons between adjacent layers are all pairwise connected. The output layer neurons dont have an activation function, and their job is to represent the class scores (classification) or some kind of real-valued target (regression). Two metrics are commonly used to measure the size of neural networks are the number of layers and the number of parameters (weights). Modern Convolutional Networks contains on orders of 100 million parameters and are usually made up of approximately 10-20 layers.

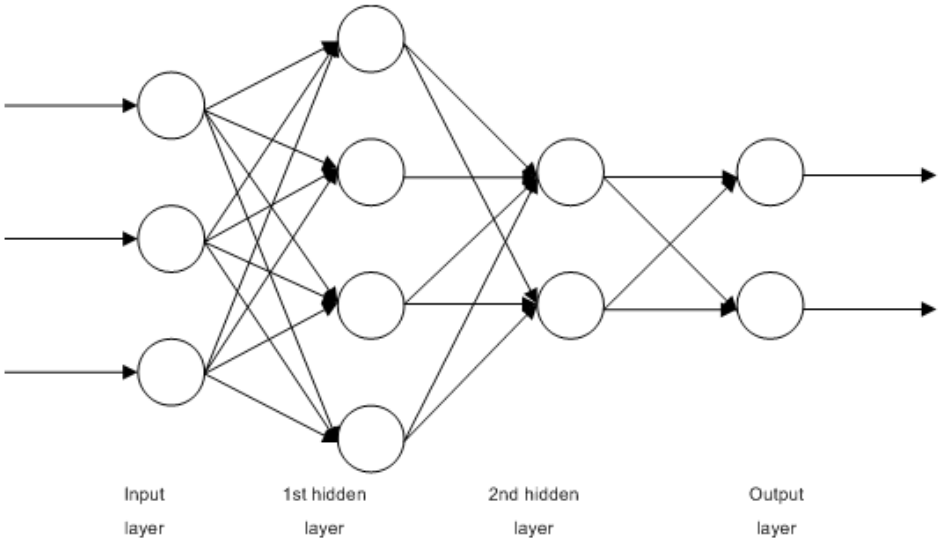

Figure 3.5: A three layers neural networks, with three inputs, two hidden layers of four and 2 neurons and one output layer. Neurons between two layers are fully connected, but no connection within a layer.

By organizing neural networks into layers, we can simply and efficient to evaluate neural networks using matrix vector operations. In a three-layer neural networks with input as a $[5 \times 1]$ vector, the first hidden layers weights W_1 can be a size of $[4 \times 5]$ matrix, and the biases for all units would be a $[5 \times 1]$ vector. The processing with in the layer is a simple production of W_1 and data vector. Similarly, The second layers weights W_2 can be a $[3 \times 5]$ matrix. And the output layer can be a matrix $[2 \times 3]$ for a two classes classification problem. Then the three-layer neural networks is then simply three matrix multiplications:

$$W_3 \times f_2(0, W_2 \times f_1(W_1 X)) \tag{3.1}$$

Where f_1, f_2 are activation function for layer 1 and layer 2.

Fully-connected neural networks can be seen as a family of functions that are parameterized by the weights of the networks. And neural networks with at least one hidden layer is universal approximator [6]. This means that given any continuous function $f(x)$ and some $\epsilon > 0$, there exists a neural network $g(x)$ with one hidden layer such that $\forall, |f(x) - g(x)| < \epsilon$.

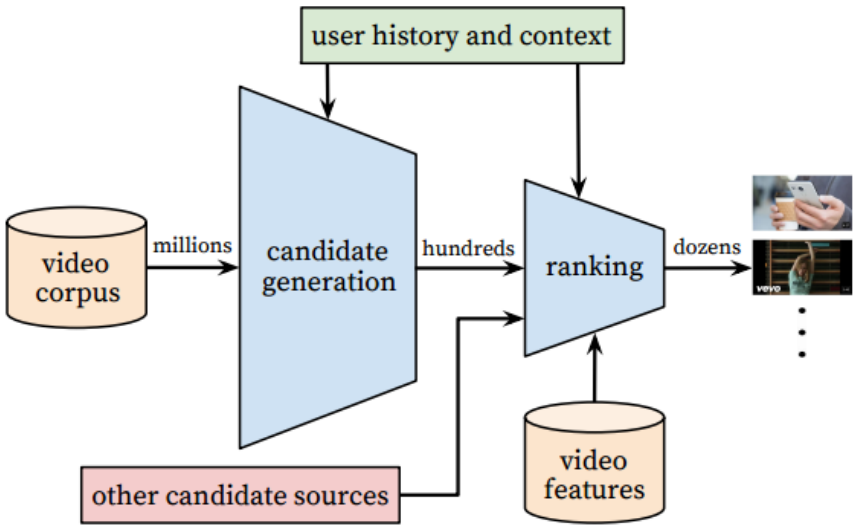

Figure 3.6: The two-stage approach to recommendation allow YouTube to make recommendations from a very large corpus of videos.

3.3 Neural networks for YouTube Recommendations

YouTube has the worlds largest platform for creating, sharing and discovering video content. The recommendation system in YouTube are responsible for helping more than a billion users discover personalized content from an ever-growing corpus of videos.

The overall structure of recommendation system in YouTube is comprised of two neural networks: on for candidate generation and one for ranking [5]. As shown in Figure2 the system concatenates these two networks together as a whole system.

The candidate generation network takes events from the users YouTube video history as input data. Through the neural network, find a small subset of videos from a large corpus. These candidates are intended to be a general set contains videos that are closely relevant to the user. Then the candidate generation network only provides broad personalization via collaborative filtering.

The ranking network was designed to assign a score to each video according to a desired objective function using a rich set of features describing the video and user. The highest scoring videos are presented to the user, ranked by their score.

Hidden layers	Weighted per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU → 256 ReLU	34.7%
1024 ReLU → 256 ReLU → 256 ReLU	34.6%

Table 3.1: Effects of different layers on watch time-weighted pairwise cross entropy loss computed on next-day holdout data.

3.4 Recurrent neural networks based recommendation system

Besides learning directly from ratings of items, some methods [12] learn users' preference from their review contents. The large repositories of user written reviews create opportunities for a new type of recommendation system that can leverage the rich content embedded in the written text.

To create the input to RNN models, each word in the review will be converted into distributed representation in the form of word vector. Each word vector in the review serves as input to a hidden layer of the RNN [13]. The output is a prediction of the probability that user will like the certain item with input historical review on other items.

The RNN can be an extremely expressive model that learns highly complex relationships, while the depth of RNN often leads to vanish gradient problem [14][15]. Both LSTM and GRU can solve the vanishing gradient problem by reparameterizing the RNN.

The recurrent neural networks are said to have higher effective in predicting user preference base on their reviews, the multi-stack bidirectional RNN (BiRNN) can produce more accurate prediction compared to traditional content based recommendation systems [12]

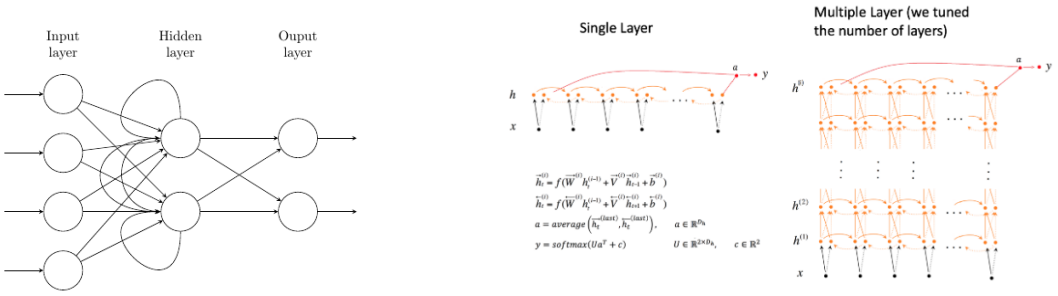

Figure 3.7: **Left**: a recurrent neural network with 1 hidden layer. **Right**: one of the recurrent neural networks based recommendation system BiRNN, it consists of forward and backward RNN structure.

3.5 Music recommendation with deep learning

Traditional Spotify recommendation system relied mostly on collaborative filtering approaches. The underlying idea of deep learning approach is that many collaborative filtering models work by projecting both the listeners and the songs into a shared low-dimensional latent space [9].

The neural networks that Spotify trained for recommendation system consisted of two convolutional layers and two fully connected layers. And the input data is spectrograms of three-second fragments of audio.

In this recommendation system, neural network is supposed to detect music style by learning various filters for different kinds of harmonics. Every layer in the network takes the output signals from the layer below, and extract features from them. At the topmost fully-connected layer of the network, the learned filter can be very selective for certain subgenres [9].

3.6 Neural network matrix factorization

In the matrix factorization approaches in recommendation system, the rating matrix is approximated by the inner product of user matrix U and item matrix V. The inner product can

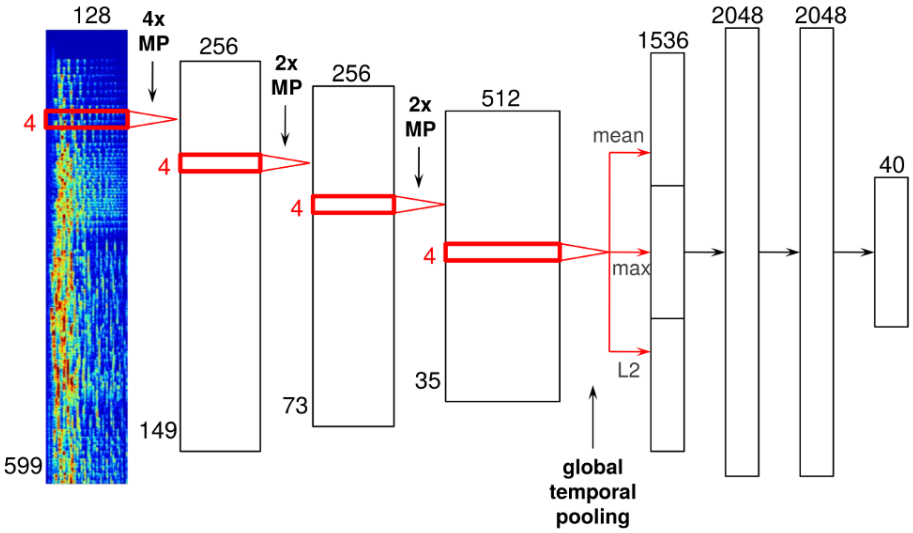

Figure 3.8: The architecture of Spotify neural network, which has two convolutional layers and two fully connected layers. The output layer predicts 40 latent factors which can be obtained from previous collaborative filtering algorithms that are used in Spotify. The network is trained to minimize the mean square error.

be seen as a simple fixed function $f(U, V) = U \times V$. The neural network matrix factorization (NNMF) based on the idea that $f = f_\theta$ is a feed-forward neural network with weights θ [7]. Given data, NNMF will learn weights θ as well as the latent features U and V.

To learn the network weights θ and latent features (U, V), the loss function in NNMF is defined as:

$$\sum_{(n,m)\in J} (X_{n,m} - \hat{X}_{n,m})^2 + \lambda(\sum_n \|U'_n\|_F^2 + \sum_n \|U_n\|_2^2 + \sum_m \|V'_m\|_F^2 + \sum_m \|V_m\|_2^2) \tag{3.2}$$

Where $J \subset N \times M$ are the indices of those observed data (rated items), λ is a regularization parameter for I_2 norm, and $\|\cdot\|_F$ is the Frobenius norm for latent item feature. During the training, the algorithm alternated between optimizing the neural network weights, while fixing the latent features, and optimizing the latent features, while fixing the network weights. Optimization was carried out by gradient descent on the entire dataset without using batches.

In the experiments of NNMF, it dominated other previous latent feature methods in the data: MovieLens 100K and 1M. But NNMF was dominated by both LLORMA and I-AutoRec. The possible reason for the difference is the limitation of latent feature models, which assume that the ratings are conditionally independent given the latent feature representations [7]. Some recent work demonstrates that explicitly modeling the non-random pattern of missing ratings can lead to a slight improvement in performance for latent feature models [8]. Therefore, if a model can predict the position of these songs position in that space, Spotify can recommend them to neighbor listeners.

CHAPTER 4

Application of recommendation system to MovieLens

4.1 Introduction to the MovieLens dataset

In this project I use the movie ratings dataset released by MovieLens `https://grouplens.org/datasets/movielens/`.It is a dataset collected by the GroupLens research project for research-minded academics at the University of Minnesota. The data set was collected through the MovieLens web site, and has been cleaned up users who had less than 20 ratings. In this paper I focus on data sets with 100,000 (100k) and 100,000,000 (1M) ratings (1-5). Table 3 summarizes the properties of the dataset.

The data was stored separately in three different ".dat" files, which are ratings, users and movies. In this paper, only the rating.dat file that is most relevant is use. All movie ratings are contained in this file and are in the format "UserID::MovieID::Rating::Timestamp", where ratings are made on a 5-star scale (whole-star ratings only) and each user has at least 20 ratings. In this project we developed Python code to read rating data into pandas DataFrame object, and randomly split the data into training and testing dataset. We introduce Pytorch to help us construct the neural network.

	100K data set	1M data set
Number of ratings	100,000	1,000,209
Number of Users	943	6,040
Number of Movies	1,682	6,040

Table 4.1: The properties of MovieLens datasets.

4.2 Bi-generator neural network for recommendation system

Traditional latent factor methods assume that the ratings of items are fixed as well as characteristics (latent factors) of users U and items V. In the Bi-generator model, we describe a different approach to factorizing R. Our idea is to learn the rating as a random variable r that follows a distribution $f(\cdot)$, rather than assume it is fixed. In particular, we take $p(\cdot)$ as the probability from not like the item at all (0) to like (1). The major challenge for this model is to learn the $f(\cdot)$, which can be a very function.

4.2.1 Model

To simplify the problem, we can first start from a uniform distribution, that is $r \sim U(0, 1)$. In this system, users randomly rate movies without any bias and preference. Since the uniform distribution is too naive to approximate the target function $f(\cdot)$, we can use a more complex distribution, Gaussian distribution, to approximate $f(\cdot)$, that is $r \sim N(0, 1)$.

For a particular rating $r_{i,j}$ from user i rate item j, we can add preferences and bias to let the distribution of r_ij be more flexible and precise. Then we rewrite the distribution $r_ij \sim N(\mu_ij, \sigma_ij)$. We can relax the assumption further by letting r to follow an arbitrary distribution $f_ij(\cdot)$. Since i and j can be interpreted as the characteristic of users and items, we can generalize all $f_ij(\cdot)$ into a distribution family $f(u, v)$. For a particular i and j, we have $r_ij \sim f(u_i, v_j)$.

Let p_ij be the probability that user i likes item j, then $p_ij = f(u_i, v_j)$. Since costumers usually don't involve in the design and product of any item, it is reasonable to assume that u and v are independent. Therefore we can write the formula as $p_ij = f_u(u_i)f_v(v_j)$, and f_u, f_v can be seen as function take parameters u_i, v_j. In the original matrix factorization, f_u, f_v are simple linear function: $f(x) = x$. The idea is to learn f_u, f_v by taking $f_u = f_{\theta,u}, f_v = f_{\eta,v}$ to be two feed-forward neural networks with weights θ, η.

In the model, f_θ, f_η are two feed-forward neural networks that can transform simple distribution (uniform) to more complex distribution. These two neural networks have weights

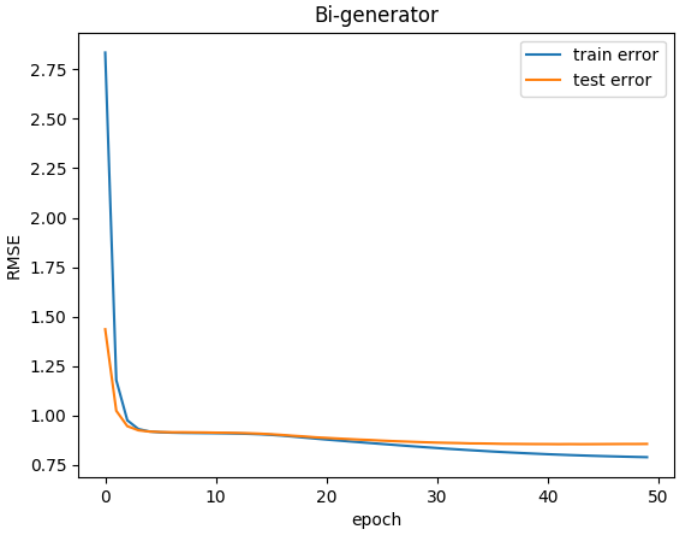

Figure 4.1: The testing error and training error of one layer Bi-generator neural network in 100K data with K = 15, batch size 500.

θ, η, which can be learned by stochastic gradient descent.

4.2.2 Learning

To learn the network weights θ and η we minimize the objective

$$\sum_{u,v}(p_ij - \hat{p_ij})^2 + \lambda_1\|\theta\|_2^2 + \lambda_2\|\eta\|_2^2 \tag{4.1}$$

Where $\hat{p}_ij = f_{\theta,u}(u_i)f_{\eta,v}(v_j)$, and λ_1, λ_2 are regularization parameters. This can be understood as a penalized log likelihood under a Gaussian model for every observation. Also, since $\hat{p}_ij = p(u_i, \theta)p(v_j, \eta) = p(u_i|\theta)p(v_j|\eta)p(\theta)p(\eta)$, it can also be understood as a specifying the maximum a posteriori estimation.

To generate complex distribution, we start from two one-dimensional uniform distributions. The generated distributions have K dimension, where K can be interpreted as the number categories of movies. Therefore, the neural network we built has one input and K output. During the training, we simultaneously optimizing the neural network weights θ and η. Optimization was carried out by gradient descent on a batch of dataset.

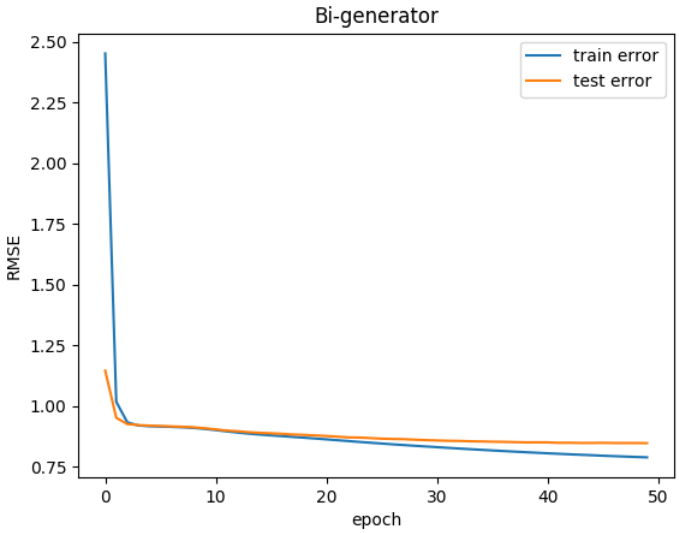

Figure 4.2: The testing error and training error of one layer Bi-generator neural network in 100K with K = 15, batch size 1000.

CHAPTER 5

Model validation

The most commonly used measurement to evaluate a recommendation system is their prediction power, i.e. ability to accurately predict the user's choices. One of the choice is to perform off line experiments using existing data set [16]. A more expensive option is a user study, where a number of users will be asked to perform a set of operations using the system, then answering questions afterwards about their experience. In this paper, I used off line experiment to validate the efficiency of the matrix factorization, neural network matrix factorization and Bi-generator recommendation systems. The candidate recommender systems can be ranked with respect to these properties. The properties include root mean squared error (RMSE), precision and recall, area under the ROC curve and Normalized Distance-base Performance Measure (NDPM).

I applied the matrix factorization and Bi-generator matrix factorization on the MovieLens dataset, and use RMSE to measure their performance

Data set	MF	NNMF	Bi-generator MF
100K	0.910	0.903	0.910
1M	0.858	0.843	0.860

Table 5.1: The performance of three different recommendation systems, the measurement in the table is RMSE. NNMF here takes the performance of 3 hidden layer neural network, and the data was from the original paper [7].

CHAPTER 6

Conclusion

This paper explores the application of a different neural network based recommendation system, Bi-generator neural network. The neural network based systems have been proved to be more accurate than traditional recommendation systems in on-line applications, such as YouTube and Spotify[5]. This paper made attempt to apply the Bi-generator recommendation system, which uses the ability of neural network to express arbitrary distributions. In the experiment, we find the Bi-generator model can generate a complex distribution, but it's prediction is dominated by traditional matrix factorization and neural network matrix factorization. Consider Bi-generator usually has more parameters to train, hence it's more likely to suffer from overfitting than other methods. Because of time limit and the limit in the computational efficiency of my PC, I applied Bi-generator with no more than 2 hidden layers. It's possible that my experiments have not identified the limits of Bi-generator model. The neural network techniques allow us to build more flexible and efficient models and hence improve the recommendation results. It would be interesting to apply Bi-generator model with more hidden layers, and to dataset of a larger scale.

REFERENCES

[1] Jure Leskovec, Anand Rajaraman, Jeff Ullman. *Mining of Massive Datasets*. Cambridge University Press, November 2014.

[2] Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, Stefan Reiterer, Martin Stettinger. *Basic Approaches in Recommendation Systems*. Recommendation Systems in Software Engineering, December 2013.

[3] Ya-Yueh Shih, Duen-Ren Liu. *Hybrid recommendation approaches: collaborative filtering via valuable content information* pp. 217p-217b IEEE. 2005.

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks* pp. 1097-1105.

[5] Paul Covington, Jay Adams, Emre Sargin. *Deep Neural Networks for YouTube Recommendations* pp. 191-198.

[6] Michael Nielsen. *Approximation by Superpositions of a Sigmoidal Function* (MCSS), 2(4), 303-314.

[7] Gintare Karolina Dziugaite, Daniel M. Roy. *Neural Network Matrix Factorization* arXiv: 1511.06443.

[8] J. M. Hernandez-Lobato, N. Houlsby, and Z. Ghahramani. *Probabilistic Matrix Factorization with Non-random Missing Data* pp. 1512-1520.

[9] Arron Vandenoord, Sander Dieleman, Benjamin Schrauwen. *Recommending Music on Spotify with Deep Learning* pp. 2643-2651

[10] Robert Bell, Chris Volinsky. *Matrix Factorization Techniques for Recommender Systems* Computer 42.8 (2009)

[11] Jon Herlocker, Joseph A Konstan, and John Riedl. *An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms* 5(4), 287-310.

[12] David Zhan Liu, Gurbir Singh *A Recurrent Neural Network Based Recommendation System*

[13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. *Distributed Representation of Words and Phrases and their Compositionality.* NIPS 2013

[14] Bengio, Yoshua, Simard, Patrice, Frasconi, Paolo *Learning long-term dependencies with gradient descent is difficult* IEEE Transactions on, 5, pp. 157-166.

[15] Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. *An empirical exploration of recurrent network architectures* ICML-15, pp. 2342-2350, 2015

[16] Guy Shani and Asela Gunawardana. *Evaluating recommendation systems.* In *Recommender systems handbook* pp. 257-297. Springer, 2011.

[17] Michael J. Pazza Ni *A Framework for Collaborative, Content-Based and Demographic Filtering* Artificial intelligence review, 1999, 13(5-6): 393-408.

[18] Van Meteren, Robin, Maarten Van Someren. *Using content-based filtering for recommendation.* Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop. 2000.

[19] Pazzani, Michael, and Daniel Billsus. *Content-based recommendation systems.* The adaptive web (2007): 325-341.

[20] Howe, Michael *Pandoras Music Recommender.* A Case Study, I (2009): 1-6.