Winter 2018

Deep Learning based Recommendation Systems

Nishanth Reddy Pinnapareddy
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Pinnapareddy, Nishanth Reddy, "Deep Learning based Recommendation Systems" (2018). *Master's Projects*. 644.

DOI: <https://doi.org/10.31979/etd.k9vm-pxcs>
https://scholarworks.sjsu.edu/etd_projects/644

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Deep Learning based Recommendation Systems

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Nishanth Reddy Pinnapareddy

May 2018

© 2018

Nishanth Reddy Pinnapareddy

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Deep Learning based Recommendation Systems

by

Nishanth Reddy Pinnapareddy

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2018

Katerina Potika	Department of Computer Science
-----------------	--------------------------------

Sami Khuri	Department of Computer Science
------------	--------------------------------

Abhinand Lingareddy	VMware Inc.
---------------------	-------------

ABSTRACT

Deep Learning based Recommendation Systems

by Nishanth Reddy Pinnapareddy

The usage of Internet applications, such as social networking and e-commerce is increasing exponentially, which leads to an increased offered content. Recommender systems help users filter out relevant content from a large pool of available content. The recommender systems play a vital role in today's internet applications. Collaborative Filtering (CF) is one of the popular technique used to design recommendation systems. This technique recommends new content to users based on preferences that the user and similar users have. However, there are some shortcomings to current CF techniques, which affects negatively the performance of the recommendation models. In recent years, deep learning has achieved great success in natural language processing, computer vision and speech recognition. However, the use of deep learning in recommendation domain is relatively new. In this work, we tackle the shortcomings of collaborative filtering by using deep neural network techniques.

Although some recent work has employed deep learning for recommendation, they only focused on modeling content descriptions, such as content information of items and auricular features of audios. Moreover, these models ignore the important factor of collaborative filtering, that is the user-item interaction function, but some models still employ matrix factorization, by using inner product on the latent features of items and users.

In this project, the inner product is replaced by a neural network architecture, which learns an user-item interaction function from data. To handle any non-linearities in the user-item interaction function, a multi-layer perceptron is used.

Extensive experiments on two real-world datasets demonstrate improvements made by our model compared to existing popular collaborative filtering techniques. Empirical evidence shows deep learning based recommendation models have better performance.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Prof. Katerina Potika, who expertly guided me through my graduate education and my master's project. Her unwavering enthusiasm for the study of social networks kept me engaged with my research. Her constant mentorship, advice and support helped me to move in a right direction towards completion of the project. I would like to thank her for her time, help and efforts towards me and this project.

My deep gratitude also goes to Prof. Sami Khuri and my co-worker at work Abhinand Lingareddy for being on my defense committee. I would like to thank them for their time and efforts. Lastly, I would like to thank my friends and family. They supported and helped me to survive this stress and not letting me give up.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Background	4
2.1	Recommender Systems	4
2.1.1	Content-based filter techniques	5
2.1.2	Collaborative filter techniques	5
2.1.3	Hybrid filtering techniques	6
2.2	Deep Learning and Artificial Neural Networks	7
2.2.1	Feedforward Neural Network	8
3	Problem Definition	10
3.1	Recommendations from Implicit Feedback	10
3.2	Matrix Factorization	11
4	Related Work	13
5	The Model	16
5.1	General Framework	16
5.1.1	Learning Model Parameters	18
5.2	Generalized Matrix Factorization (GMF)	18
5.3	Multi-Layer Perceptron (MLP)	19
5.4	Neural Matrix Factorization	20
6	Experimental Results	22
6.1	Datasets	22

6.1.1	MovieLens	22
6.1.2	Pinterest	23
6.2	Evaluation Metrics	23
6.3	Competing Methods	24
6.3.1	Most-Popular Item	24
6.3.2	User-KNN	24
6.3.3	Bayesian Personalized Ranking	24
6.4	System Configuration	25
6.5	Parameter Settings	25
6.6	Performance Comparisons	26
6.6.1	Experiments - Research Question 1	26
6.6.2	Experiments - Research Question 2	31
7	The Conclusion and Future Work	33
	LIST OF REFERENCES	34

LIST OF TABLES

1	Characteristics of Datasets	22
2	System Configuration	25
3	Parameter variations	25
4	Neural MF performance with and with out pre-training	31
5	Hit Ratio@10 of MLP with different hidden layer units	32
6	NDCG@10 of MLP with different hidden layer units	32

LIST OF FIGURES

1	Recommender System techniques	4
2	Representation of neuron	7
3	An example explains MF’s limitation	12
4	Generalized Neural Network Framework	17
5	Neural Network Matrix Factorization	20
6	Performance of HitRatio@10 and NDCG@10 on MovieLens Dataset	27
7	Performance of HitRatio@10 and NDCG@10 on Pinterest Dataset	28
8	Top-K item recommendation on MovieLens Dataset	29
9	Top-K item recommendation on on Pinterest Dataset	30

CHAPTER 1

Introduction

Recommender systems are intelligent systems which exploit users ratings on items in the past to recommend similar items to other users. These systems play a crucial role in on-line businesses by pro-actively narrowing the navigation items for the users based on their preferences. The main problem solved by recommender systems is the problem of information overload. For bigger companies, having an efficient recommender system would provide a positive impact on their revenue. The personalized content provided by a recommender system would improve the experience of the user which would save them a lot of time.

Collaborative filtering (CF) [1], Content-based filtering (CB) and Hybrid Methods are the popular recommendation techniques in recommender system domain. The collaborative filtering methods and hybrid methods use different criteria to suggest the items tailored for users. For example, CF-based methods make use of history of the user ratings on products whereas hybrid methods, which combine both collaborative filtering and content-based methods. Due to some security concerns in CB methods like collecting user profile information, collaborative filtering based methods became popular for personalized recommendations. Matrix Factorization (MF) [2] is the most popular method of all CF based methods. This method rely on user-item interactions, where interactions can be represented or modeled as inner product of latent vectors.

After Netflix Prize [3] competition, Matrix Factorization became the default method to model user-item interactions based collaborative filtering methods. A lot

of research work was put in to enhance Matrix Factorization, essentially to integrate MF along with neighbor-based models [4], merging MF with topic models of the item description [5], and extending the functionality to factorization machines [6] for analyzing general methods to model the features. Though MF is very effective for collaborative filtering, it is also a known fact that its capabilities could be negatively impacted by selecting the inner product of IF(Interaction Function). Consider a scenario of rating the analysis on EF(Explicit Feedback), where we very well know that by using item bias terms and user into IF, it will improve the performance of MF model. Interactions between items and users which are termed as latent feature interactions can be designed in an effective way by just making a minor tweak to the inner product operator. Inner product, which is a combination of product of latent features in a linear method, would not be sufficient for obtaining the complex model of the data about user interaction.

Many people have previously worked on handcraft models but we have explored the IF from data instead of handcraft by using deep neural networks [7]. Several domains like processing the text from speech recognition and computer vision [8] are the prominent areas where deep neural networks[DNN's] have already proven their capability of calculating any continuous function(CF) [9]. There is a large amount of literature available on MF models but a considerably lesser work has been done to get recommendation by applying DNN's. Some latest advancements [10] have used DNN's for recommendations and have reported great results. DNN's have been mostly used to model auxiliary information like visual content of images, textual information on items and audio features of music. But to model the effect of collaborative filtering, MF has been used for the calculation of inner product by adding user and latent features of an item.

In this project, we address the drawbacks of Matrix factorization by replacing inner product with a neural network architecture. Rather than using explicit ratings we focused on implicit ratings, that inherently shows user preference through behaviors like clicking or buying items and watching videos and can be tracked automatically. Thus, it makes easy for content providers to collect data. However, there is one problem with this feedback as we can not differentiate between positive feedback and negative feedback. To reduce the effect of negative feedback we utilize deep neural networks to model recommendation task. The main contribution of our work is that we extended the existing approaches of Generalized Matrix Factorization(GMF) and Multi-layer Perceptron(MLP) models [11] by fusing them together to a new model Neural Matrix Factorization and performed extensive experiments to evaluate the performance of this new model.

CHAPTER 2

Background

This chapter presents the essential background.

2.1 Recommender Systems

A recommender system or a recommendation system (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

Figure 1: Recommender System techniques

As shown in Figure 1, the techniques in recommender systems is broadly classified into three categories.

- Collaboratives filtering techniques rely on user activity like ratings on items or buying patterns.
- Content based filtering techniques rely on user activity attributes like keywords

during search or their profiles.

- Hybrid filtering techniques combine both above techniques to overcome their limitations and improves performance.

The following sections we will cover each of the techniques and their limitations.

2.1.1 Content-based filter techniques

These methods use both content description or descriptive attributes of items and user activity to make recommendations. Content-based methods works better for new items in the system since they find the similar items based on items descriptive attributes which are rated by active user. However, these techniques restrict models to predict particular type of items as they rely on keywords and content of items. They also don't work well when making predictions for new users.

2.1.2 Collaborative filter techniques

Models based on this technique rely on the collaborative power of ratings provided by users. The key challenge in building these models is that ratings matrices are sparse. Collaborative filtering methods predict unspecified ratings based on the observed ratings since they are often highly correlated across various users and items. Memory and Model based methods are commonly used techniques.

2.1.2.1 Memory based methods

These methods are also referred as neighborhood-based collaborative filtering algorithms. Here, the predictions are based on the neighborhoods. These neighborhoods can be based on one of the below.

- **User-based collaborative filtering:** This filtering technique will give recommendations based on the ratings provided by like-minded people. For example, if you want to provide suggestion to user A you determine all the users who are similar to user A and recommend ratings for the missing ratings of A by calculating the weighted averages of the ratings of similar set of users.
- **Item-based collaborative filtering:** Here, to predict the ratings for target item B by user A, we need to calculate the set of items which are similar to item B. The ratings in this set provided by A will help us determine if user A will like item B, or not.

Memory based models are simple to implement and easy to understand. However, they don't perform well with sparse rating matrices that is they lack full coverage of rating predictions. Nevertheless, this won't be an when we want to predict top-k items.

2.1.2.2 Model based methods

These methods use predictive data mining and machine learning techniques to make recommendations. In case of parametrized models, these parameters are learned using optimization frameworks. Decision trees, Rule-based models, Bayesian methods and latent factor models are some examples. These models have high level of prediction coverage even for sparse ratings. However, these methods tend to be heuristic and don't perform well under all settings.

2.1.3 Hybrid filtering techniques

This is the best recommender system when we have diverse set of input categories in dataset. Various aspects from the above-mentioned recommender systems can be

used to achieve the best results. This will use the power of many machine learning algorithms in a combined way to create a robust model.

2.2 Deep Learning and Artificial Neural Networks

Deep learning is a subset of Machine Learning family, which learns data representations, as opposed to task specific algorithms. Deep learning models use a cascade of multi layered non-linear processing units called as neurons, which can perform feature extraction and transformation automatically. The network of such neurons is called an Artificial Neural Network.

Artificial Neural Network is a computational model that is inspired by the way biological neural networks in the human brain process information. The smallest unit of computation in neural network is neuron, often called as a node or unit. It receives inputs from other neurons and computes an output. Each input to the node has a associated weight (w) which signifies its relative importance to other inputs. The node applies function f to weighted sum of inputs as shown in Figure 2:

Figure 2: Representation of neuron
[12]

The function f is non-linear and often referred to as Activation function. This

function is useful to learn complex patterns in data.

2.2.1 Feedforward Neural Network

This neural network contains multiple neurons arranged in layers. Neurons from adjacent layers have connections between them and each of these connections have weights associated with them. Figure 3 shows an example of feedforward neural network.

Figure. 3. Feedforward neural network [12]

Feedforward neural network consists of three types of nodes:

- **Input Nodes** - These nodes provide information from external sources to network and together it is referred as "Input Layer". These nodes don't perform any computation, they just pass the information to hidden nodes.
- **Hidden Nodes** - These nodes does not have any connection with external world. They perform computations and transfer information from input to

output nodes. Feedforward network will have only single input and output layers, but it can have zero or more hidden layers.

- **Output Nodes** - These nodes perform computations and transfer information to outside world. The collection of Output nodes referred to as "Output Layer".

In feedforward networks, the information moves along only in one direction - forward - from input nodes, through hidden nodes and finally to output nodes. It does not contain any cycles or loops. **Single Layer Perceptron** is type of feedforward neural network does not contain any hidden layer. **Multi Layer Perceptron** has one or more hidden layers.

CHAPTER 3

Problem Definition

In this chapter we introduce the problem and then discuss about existing collaborative filtering techniques based on implicit data. We then discuss the most popularly used Matrix Factorization method and its limitation due to inner product of user and item latent vectors.

3.1 Recommendations from Implicit Feedback

Let M and N represent the number of users and items respectively. We denote interaction matrix between user and items as $\mathbf{Y} \in \mathbb{R}^{M \times N}$.

$$y_{ui} = \begin{cases} 1, & \text{if there is an interaction between user and item.} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

If the value of y_{ui} is 1, then is an interaction between user and item. Otherwise there is no interaction between user and item. Since, these interactions do not specify whether actually the user likes the item or not there is possibility of noise signals.

We formulate predicting recommendations from implicit data as predicting the scores of unobserved entries in $\mathbf{Y}$. Model based techniques abstracts learning as $y_{ui} = f(u, i | \Theta)$, where $\hat{y}_{ui}$ represents the predicted score for y_{ui} , Θ represents model parameters and f represents a interaction function that maps model parameters to calculated score.

Existing techniques use machine learning algorithms, which optimize objective function to calculate model parameters Θ . These techniques commonly use two types

of objective functions - pointwise loss [12] and pairwise loss. Most models use pointwise loss objective function and they learn by following a regression framework by minimizing the squared loss between predicted score ($\hat{y}_{ui}$) and observed score (y_{ui}). These models handle negative feedback either by sampling negative entries from observed entries [13] or by treating all unobserved entries as negative feedback. Models based on pairwise loss function assumes the observed entries should be ranked higher than the unobserved entries. These models increase the difference between observed entry $\hat{y}_{ui}$ and unobserved entry $\hat{y}_{uj}$. Our proposed model based on neural network supports uses pointwise learning, but it can also be extended to pairwise learning.

3.2 Matrix Factorization

Matrix Factorization (MF) is one of the most popular collaborative filter techniques used in Industry for Recommender Systems. It pairs each user-item interaction with a real valued vector of latent features. Let p_u and q_i represent the real valued latent vectors for user and item, respectively. MF computes (y_{ui}) as the dot product of p_u and q_i :

$$\hat{y}_{ui} = f(u, i | \mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^K p_{uk} q_{ki} \quad (2)$$

where K represents latent space. The bi-directional communication of user and product latent factors considering the each direction of latent space is not connected with each other and linearly adding them with same load. Therefore, MF can be considered as a one dimensional model of latent factors.

To understand the above illustration well, there are two settings to be stated clearly beforehand. Since the latent space is the result of mapping users and products in the same dimension then, the dot product or the cosine of angle in latent vectors gives us the similarity between two people. The second point is the Jaccard coefficient [14]

Figure 3: An example explains MF’s limitation

[14] From user-item matrix (a), u_4 is most identical to u_1 , followed by u_3 , and lastly u_2 . However, in user latent space (b), placing p_4 closet to p_1 makes p_4 closer to p_2 than p_3 , resulting greater ranking loss.

which helps MF to calculate similarity between the users without losing generality between them.

We can see from the first three rows of user-item matrix in Fig. 3a, the cosine similarity score of $s_{23}(0.66) > s_{12}(0.5) > s_{13}(0.4)$. As such, the geometric relations of p_1, p_2 , and p_3 in the latent space can be plotted as in Figure 3b. Now, let us consider a new user u_4 , whose input is represented by dashed line in Fig. 3a. We can have $s_{41}(0.6) > s_{43}(0.4) > s_{42}(0.2)$, meaning that u_4 is most similar to u_1 , followed by u_3 , and lastly u_2 . However, if p_4 is placed closer to p_1 by this model, it will result in p_4 closer to p_2 than p_3 , which unfortunately will results in greater ranking loss.

From this illustration, we can see the negative impact created by simple and fixed inner product on model performance. Our models address this drawback by learning user-item interactions using deep neural networks which is covered in later sections.

CHAPTER 4

Related Work

Past models rely on data from explicit feedback as the primary source for recommendations tasks [15], but the attention is slowly moving towards implicit data. The implicit feedback of collaborative filtering is usually interpreted as a problem of recommendation of the item that focuses on recommending a simple item list for users. The problem on predicting the rating is broadly solved so far by the work done on explicit feedback(EF) but it is more practical to solve the problem on item recommendation but it is more challenging. To design the models of latent factor for the item recommendation based on implicit feedback(IF) , recent works added to a uniform weighting where proposal is made with two strategies, which considered all the data missing to be negative instances or derived the negative instances from the data that was missing. To weigh the missing data, dedicated models have been proposed by He et al [2] and Liang et al [16]. For the models that are based on feature based factorization, Rendle et al [17] implemented an implicit coordinate descent (iCD), which achieved the cutting-edge performance for recommendation of the item. Neural networks usage for the recommendation works is discussed in depth in the following content.

The work done by Salakhutdenov et al. [15] involves a two layered Restricted Boltzmann Machines for modeling the users that contain explicit ratings for the items. This particular work was then extended to model the ratings for ordinal nature[ref]. In recent times, the mostly used choice to build the recommendation systems is autoencoders. A study of hidden patterns that are capable of reconstructing the ratings of a user with the inputs of historical ratings is called user-based AutoRec [18]. Rather

than personalizing the user data this approach is shares a similarity with item-item model[ref] where the rated items represent a user. For the purpose of avoiding autoencoders identity function learning and failure to generalize the unseen data, the introduction of denoising autoencoders (DAE's) has been done to study from the inputs which are intentionally corrupted[ref]. A neural autoregressive method for collaborative filtering (CF) has been recently proposed by Zheng et al [19]. The effort which has been put previously has provided a very strong support which improved the success of neural networks (NN) to address the problem of collaborative filtering where the focus was more on the explicit ratings and it is only modeled using observed data. Accordingly, they could fail in learning users preferences because implicit data is positive.

While some recent work [20] have analyzed recommendation established on implicit feedback (IF) by using deep learning models, they have mainly used deep neural networks (DNN's) to model the additional information like text description of the items, sound properties of the music which deals with physics, behavior of users across multiple domains, and abundant content in the knowledge areas. These particular features derived from deep neural networks are then combined with Matrix Factorization for collaborative filtering. The one which is more similar to the work [21], that ensures the auto-encoder of collaborative denoising also termed as (CDAE) for collaborative filtering with the implicit feedback (IF). Contrary to the denoising auto-encoder based collaborative filtering, collaborative denoising autoencoder(CDAE) also pushes a node of user into autoencoders(AE) input for reconstruction of ratings of users. According to these authors, collaborative denoising autoencoder(CDAE) shares some similarities with SVD++ model [20] where the activation of hidden structures of collaborative denoising autoencoder(CDAE) can be obtained by the application of identity func-

tion. Though CDAE is used as neural modeling method for collaborative filtering , it also involves applying inner product to design or model the user and item interactions(UII). This explains very well why the usage of deep layers for collaborative denoising autoencoder will not enhance its performing ability (cf. Section 6 of [21]). Noticing this typical behavior from collaborative denoising autoencoder the NCF portrays a two - way architecture where the user and item interactions are modeled with multi-layer feedforward neural network(MFNN). This helps NCF to derive a function which is arbitrary from the data provided which is more self-explanatory and very much capable than the inner product function(IPF) which is constant.

Identically, grasping the relationship between two objects has been worked on extensively in the previous works of knowledge base graphs [22]. A lot of development has taken place like machine learning models which are relative [13]. An other method called Neural Tensor Network has shown robust performance as it uses the neural networks to understand the interaction between two entities which is identical to our proposal. This targets a different aspect of collaborative filtering. Since Neural MF combines the functionality of Matrix factorization with Multi-layer perceptron it appears to be leveraged from NTN but Neural MF is very dynamic and general than NTN because it allows MLP and MF to learn variable sets of embeddings.

Recently, Google published their deep neural network models which they are using for product recommendations [23]. These models used Multi-Layer Perceptron architecture, which showed promising results and also made the model generic. Although these models work on different aspects of user-item interactions, we target at analyzing deep neural networks for only CF based recommender systems. In this project, we explored the use of deep neural networks to model complex user-item interactions.

CHAPTER 5

The Model

In this chapter we first present a general framework to learn user-item interaction function using neural networks with a probabilistic model which emphasizes the implicit feedback data. We then express matrix factorization (MF) [11] as a neural network model. To explore deep neural networks for collaborative filtering, a multi-layer perceptron (MLP) [11] model is used to learn user-item interaction function. Finally, we present our neural network matrix factorization model, which is a fusion of MF and MLP models. This model gets strengths of linearity of MF and non-linearity of MLP to model user-item latent structures.

5.1 General Framework

To model user-item interaction y_{ui} we used a multi-layer representation as shown in Figure 3, where the output of one layer serves as the input to the next layer. The first input layer has two input vectors v_u^U and v_i^I that represent user u and item i . These are sparse binary vectors with one-hot encoding. After input layer, there is an embedding layer. This layer is fully connected one, that projects the sparse representation to a dense vector. The resulted user/item embedding can be viewed as the latent vector for user/item in the context of latent factor model. These embedding layers are then fed into a multi-layer neural architecture to map the latent vectors to prediction scores. We can also customize each hidden layer to discover new latent structures from user-item interactions. The final layer gives the predicted score $\hat{y}_{ui}$ and the dimension of last hidden layer determines the model's capability. We performed training by minimizing the pointwise loss between $\hat{y}_{ui}$ and its actual value y_{ui} .

Figure 4: Generalized Neural Network Framework [14]

We now formulate the neural network predictive model as

$$\hat{y}_{ui} = f(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f) \quad (3)$$

where $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$, denoting the latent factor matrix for users and items and Θ_f represents the model parameters for interaction function. Since f is defined as multi-layer neural network, it can be formulated as

$$f(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f) = \phi_{out}(\phi_X(\dots \phi_2(\phi_1(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I)) \dots)) \quad (4)$$

where ϕ_{out} and ϕ_X represent the mapping function for the output layers and X-th neural network CF layer.

5.1.1 Learning Model Parameters

Generally to learn model parameters, existing pointwise methods perform a regression task with squared loss:

$$\mathbf{L}_{sqr} = \sum_{(u,i) \in YUY^-} w_{ui} (y_{ui} - \hat{y}_{ui})^2 \quad (5)$$

where Y denotes actual observations in $\mathbf{Y}$, and Y^- denote the set on unobserved observations. While the squared loss works better on data drawn from Gaussian distribution [24] it fails to perform well on binary data $[0, 1]$. So to learn model parameters on binary data, we used a probabilistic function as the activation function for the output layer ϕ_{out} . We define the likelihood function as

$$p(Y, Y^- | P, Q, \Theta_f) = \prod_{(u,i) \in Y} \hat{y}_{ui} - \prod_{(u,j) \in Y^-} (1 - \hat{y}_{uj}) \quad (6)$$

by taking the negative logarithm of the likelihood, we reach

$$L = - \sum_{(u,i) \in YUY^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log (1 - \hat{y}_{ui}) \quad (7)$$

Equation 7 is known as binary cross-entropy loss or log loss. We used this as our objective function and its optimization is performed by stochastic gradient descent (SGD).

5.2 Generalized Matrix Factorization (GMF)

In this section, we show how MF can be interpreted as a special case of neural collaborative filtering (NCF). By modeling this in to a NCF we can cover large family of factorization methods.

The input to this model is one-hot encoding of user/item vectors and then followed embedding layer can be viewed as latent vector of user/item. Let us denote

user latent vector as $\mathbf{p}_u$ and item latent vector as $\mathbf{q}_i$, respectively. We define the mapping function to first neural CF layer as

$$\phi_{out}(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i \quad (8)$$

where $\odot$ denotes the dot product of vectors. We then project the vector to output layer as:

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)) \quad (9)$$

where a_{out} and $\mathbf{h}^T$ represent activation function and edge weights of out put layer, respectively.

We implemented a generalized version of matrix factorization the uses sigmoid function as activation function and learns model parameters with log loss objective function.

5.3 Multi-Layer Perceptron (MLP)

As mentioned in section 5.1, neural collaborative filtering adopts two pathways to model user and items. It is intuitive to concatenate both these pathways [11] to design an efficient deep learning based recommender system. However, a simple vector concatenation is not enough to capture the interactions between user and item latent features. To overcome this issue, we added hidden layers on the concatenated vector, used MLP to learn the interaction between user and item latent vectors. We formulate the model as:

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(z_{L-1})) \quad (10)$$

We implemented this model with ReLU [25] as activation function and to design neural network architecture we followed a tower pattern, where the bottom is the

widest one and each successive layer has smaller number of neuron units as shown in Figure 3.

5.4 Neural Matrix Factorization

So far we have seen two neural network based models - GMF that applies linear kernel and MLP that uses a non-linear kernel, respectively to learn interaction function from data. Now, we present a hybrid model by fusing GMF and MLP so they can mutually reinforce each other and learn the complex user-item interactions.

Figure 5: Neural Network Matrix Factorization [14]

An obvious approach to fuse these models is to share both GMF and MLP same embedding layer, and then combine the outputs of their interaction functions. However, sharing embeddings of GMF and MLP may limit the performance and flexibility of fused model. So, we allowed GMF and MLP to learn separate embeddings, and

combine these models by concatenating their last hidden layers as shown in Figure 4.

We can formulate this model as:

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T(\phi_{out}^{GMF} \cdot \phi_{out}^{MLP})) \quad (11)$$

This model combines linearity from MF and non-linearity from neural networks for modeling user-item latent structures.

CHAPTER 6

Experimental Results

In this chapter, we cover the experiments that aim to answer the following research questions.

Research Question 1 - Did our proposed models out perform existing state of art collaborative filtering techniques for implicit feedback?

Research Question 2 - Are deeper hidden layers in neural network architecture beneficial to learn complex user-item interactions?

6.1 Datasets

We conducted experiments on two popularly available datasets: MovieLens and Pinterest. Table 1 shows some statistical features of these datasets.

Table 1: Characteristics of Datasets

Dataset	Interactions	Items	Users	Sparsity Percent
MovieLens	1,000,208	3,705	6,041	95.52%
Pinterest	1,500,808	9,915	55,186	99.74%

6.1.1 MovieLens

MovieLens is one of the most widely used dataset for evaluating collaborative filtering algorithms. There are different versions of this dataset available we used the one which contains 1,000,000 (million) movie ratings and every user has given more than 20 ratings. These ratings given by user are explicit, we have choose this particular dataset explicitly to evaluate the learning of implicit feedback from explicit

ratings. We converted it to implicit data by transforming each entry to 1 or 0 denoting whether user has rated the movie or not.

6.1.2 Pinterest

This implicit feedback dataset [26] is originally used for analyzing the performance of content-based image recommender systems. But, this dataset is highly sparse and more than 25% of users has only single pin which makes it harder to analyze the performance of collaborative filtering techniques. So, we modifies the dataset to be similar to Movielens dataset by ignoring users who doesn't have at-least 20 pins (interactions). Each interactions represents whether a user has pinned the image to his or her feed or not.

6.2 Evaluation Metrics

We used leave-one-out strategy to evaluate the performance of our models. According to this protocol or strategy, for each user leave out last user-item interaction which is used for testing. The remaining user-item interactions is used for training. This protocol is widely used by other implicit feedback recommendation models. Now, we need to rank the items for each user. Since it is very time consuming to do this, we used a popular strategy [23] that randomly draws top-K items, which user has not interacted and then rank the leave out item among these top-K item list. We used **Hit Ratio** (HR) and **Normalized Discounted Cumulative Gain** (NDCG) [25] evaluation metrics to measure the performance. Hit Ratio measures the whether the leave out item is present in top-K ranked list and Normalized Discounted Cumulative Gain measures the position of leave out item in the rank list by assigning high scores if it hits at high rank. We performed our experiments on top-10 item rank list for

users.

6.3 Competing Methods

We compared neural network models with following state-of-art collaborative filtering techniques. Since, our neural network models evaluates the user-items interactions, we competed our models with user-item collaborative filtering models rather than item-item models.

6.3.1 Most-Popular Item

Items are ranked by number of times they appear in user-item interactions. This model comes under non-personalized recommendation category. We used this model to baseline the recommendation model performance.

6.3.2 User-KNN

This is one of the popular user based neighborhood collaborative filtering technique [27]. We adapted this model to learn from implicit user-item interactions data by following a strategy mentioned in this [12] paper.

6.3.3 Bayesian Personalized Ranking

This model [28] is a variant of Matrix Factorization which optimizes equation present in section 3.2. using a pairwise ranking loss technique to evaluate user-item interactions from implicit feedback. This is one of the best model for item recommendations. We varied learning rate and then reported the best performance.

6.4 System Configuration

All our experiments are performed on system with the configuration in Table 2.

Table 2: System Configuration

Property	Value
Operating System	MacOS High Sierra
Processor	2.8 GHz Intel Core i7
Memory	16 GB 2133 MHz LPDDR3
Disk	512 GB SSD
Keras	1.0.8 version
Theano	0.9.0 version

6.5 Parameter Settings

We used Keras as our backend to implement neural network models. As these are parametric models, to determine these hyper parameters we randomly sample one user-item interaction for each user and used this random sample as validation data to tune hyper parameters. These models learn by optimized log loss objective function. We used Gaussian distribution, to randomly initialize model parameters especially for neural networks that are trained from scratch and then optimized using a mini-batch Adam [29]. We performed experiments by varying batch sizes and learning rates as shown in Table 3.

Table 3: Parameter variations

Property	Values
Batch Sizes	128, 256, 512 and 1024
Learning rates	0.0001, 0.0005, 0.005 and 0.001

We termed the last hidden layer in neural network model as predictive factors as it determines model capability and evaluated factors of [8, 16, 32, 64]. We used three hidden layers for Multi Layer Perceptron models, for example, if the number of

predictive factors is 16, then the architecture of neural collaborative filtering layers is 64->32->16 and the embedding layer size is 32. For our, fused neural network model with pre-training, α was set to 0.5 there by so that pre-trained Generalized Matrix Factorization and Multi Layer Perceptron models to contribute equally.

6.6 Performance Comparisons

In this section, we show how our experiments answered afore-mentioned research questions.

6.6.1 Experiments - Research Question 1

In Figures 6 and 7, we compared the performance of different models on both MovieLens and Pinterest datasets. These plots use performance metrics Hit Ratio@10 and Normalized Discounted Cumulative Gain@10 along y-axes and size of predictive factors along x-axes. For MF based Bayesian Personalized Ranking (BPR) model, the size of predictive factors matches to number of user and item latent vectors. In case of UserKNN model, we evaluated this model with different neighborhood sizes and picked the best performed one. To highlight the performance of personalized recommendation models, we ignored Most-Popular Item model.

We can seen clearly from figures 6 and 7, Neural-MF is the winner among all the competing methods and it also outperformed state-of-art collaborative filtering models by good margin. has better performance on both datasets, significantly outperformed the state of the art. On Pinterest dataset, even small number of predictive factors such as 8, 16 Neural-MF outperformed BPR model with larger predictive factors of 64. This shows the expressiveness of our model which is obtained by fusing Generalized Matrix Factorization and Multi Layer Perceptron models. We can

Figure 6: Performance of HitRatio@10 and NDCG@10 on MovieLens Dataset

also see the other neural models, Generalized MF and Multi-Layer Perceptron also have very good performance. Among them, Multi-Layer Perceptron is slightly less performed compared to Generalized MF. Generalized MF also showed significant improvements over BPR, showing the effectiveness of classification-aware log-loss for

Figure 7: Performance of HitRatio@10 and NDCG@10 on Pinterest Dataset

recommendation problems.

Figures 8 and 9 capture the performance evaluation of Top-K item recommendations where ranking position ranges from 1 to 10. To highlight the power of neural networks, we compared Neural-MF with other non-neural based methods rather than

all neural network based methods. We can see that Neural-MF shows gradual improvements compared to collaborative filtering methods. UserKNN model performed better across Model-based methods. Finally, we can see that Most-Popular Item performed the worst, indicating the importance of personalized recommendation systems.

Figure 8: Top-K item recommendation on MovieLens Dataset

Figure 9: Top-K item recommendation on on Pinterest Dataset

6.6.1.1 Use of Pre-training

To show the effectiveness of pre-training for Neural-MF model, we made comparisons between models performance with pre-training and with random initializations. For Neural-MF with random initializations we used Adam to learn model parameters.

In Table 4, we compared performance of models. In most of the cases, Neural-MF with pre-training achieves better performance compared to the one with random initializations. Thus justifying the usefulness of pre-training during initialization of Neural-MF model.

Table 4: Neural MF performance with and with out pre-training

	Pre-training model		Without pre-training	
Factors	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens Dataset				
8	0.685	0.402	0.689	0.412
16	0.708	0.427	0.697	0.421
32	0.728	0.446	0.702	0.426
64	0.732	0.448	0.706	0.427
Pinterest Dataset				
8	0.879	0.556	0.867	0.546
16	0.881	0.559	0.873	0.549
32	0.878	0.557	0.871	0.548
64	0.876	0.553	0.869	0.552

6.6.2 Experiments - Research Question 2

With less work on neural networks in recommender system domain, it is important to know whether deep neural networks are really beneficial to recommendation problems. To figure out more, we conducted experiments on MLP model by varying number of hidden layer units in MLP. The results of this experiments are shown in Table 5 and 6. The MLP@K indicate the MLP model with K number hidden layer units.

In Tables 5 and 6, we calculated the performance metrics- HR and NDCG for top-10 item recommendations on both MovieLens and Pinteret datasets. We varied the number of hidden units in MLP model from 0 to 4 and predictive factors from 8->16->32->64. We can see that increasing layers are beneficial to performance. Thus

showing the importance of deep neural layers in neural collaborative filtering models.

Table 5: Hit Ratio@10 of MLP with different hidden layer units

Factors	MLP@0	MLP@1	MLP@2	MLP@3	MLP@4
MovieLens Dataset					
8	0.453	0.627	0.656	0.672	0.677
16	0.453	0.664	0.675	0.686	0.691
32	0.454	0.683	0.687	0.692	0.699
64	0.454	0.686	0.697	0.701	0.708
Pinterest Dataset					
8	0.274	0.846	0.856	0.859	0.862
16	0.275	0.857	0.862	0.865	0.867
32	0.274	0.863	0.864	0.868	0.867
64	0.275	0.865	0.868	0.869	0.873

For MLP model with no hidden layers the performance is very less than non-personalized item popularity recommendation model. This adds values to our argument, that is simply concatenating both user and item latent vectors is not enough to model user-item interaction function and the usefulness of hidden layers.

Table 6: NDCG@10 of MLP with different hidden layer units

Factors	MLP@0	MLP@1	MLP@2	MLP@3	MLP@4
MovieLens Dataset					
8	0.254	0.358	0.383	0.399	0.406
16	0.253	0.390	0.402	0.410	0.415
32	0.251	0.407	0.410	0.425	0.423
64	0.252	0.408	0.417	0.426	0.432
Pinterest Dataset					
8	0.142	0.526	0.534	0.536	0.539
16	0.142	0.532	0.536	0.538	0.544
32	0.143	0.537	0.538	0.542	0.546
64	0.144	0.538	0.542	0.545	0.550

CHAPTER 7

The Conclusion and Future Work

In this project, we used different neural network architectures to overcome the limitations of matrix factorization collaborative filtering models. We showed these models performed better than state-of-art existing models on real world datasets. Our models are simple and generic that can be applied or extended to different types of recommendation problems. This work complements the mainstream shallow models for collaborative filtering, opening up a new avenue of research possibilities for recommendation based on deep learning.

As a future work, we want to use pairwise learners for Neural Matrix Factorization models and broaden it by using auxiliary information such user reviews, knowledge bases, and temporal signals as integral part. We want to do research in personalization models which target group of users rather than individuals. These models will be helpful in social group recommendations [30]. Apart from these models, we want to develop neural net recommender systems for multi-media products [31] which are less researched in recommendation domain. These products consists of richer visual elements that capture users interest. To add another dimension to deep neural network based models we want to explore recurrent neural networks and hashing methods [32] which further enhance the performance of recommender systems.

LIST OF REFERENCES

- [1] J. Wei, “Collaborative filtering and deep learning based recommendation system for cold start items,” *Expert Systems with Applications*, vol. 69, 2017.
- [2] X. He, H. Zhang, M. Kan, and T. Chua, “Fast matrix factorization for online recommendation with implicit feedback,” in *SIGIR*, 2016, pp. 549–558.
- [3] Netflix prize Competition, “Netflix prize competition — Wikipedia, the free encyclopedia,” 2006. [Online]. Available: https://en.wikipedia.org/wiki/Netflix_Prize
- [4] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model.” in *KDD*, 2008, pp. 426–434.
- [5] H. Wang, N. Wang, and D. Yeung, “Collaborative deep learning for recommender systems.” in *KDD*, 2015, pp. 1235–1244.
- [6] S. Rendle, “Factorization machines.” in *ICDM*, 2010, pp. 995–1000.
- [7] L. HU, “Your neighbors affect your ratings: On geographical neighborhood influence to rating prediction.”
- [8] K. H. et al., “Multilayer feedforward networks are universal approximators.” *Neural Networks*, vol. 5, pp. 359–366, 1989.
- [9] H. Z. et al., “Start from scratch: Towards automatically identifying, modeling, and naming visual attributes.” in *MM*, 2014, pp. 187–196.
- [10] F. Z. et al., “Collaborative knowledge base embedding for recommender systems.” in *KDD*, 2016, pp. 353–362.
- [11] L. He, L. Liao, H. Zhang, H. Nie, X. Hu, and T. Chua, “Neural collaborative filtering.” in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
- [12] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets.” in *ICDM*, 2008, pp. 263–272.
- [13] R. Socher, D. Chen, C. Manning, and A. Ng, “Reasoning with neural tensor networks for knowledge base completion.” in *NIPS*, 2013, pp. 926–934.

- [14] L. He, L. Liao, H. Zhang, H. Nie, X. Hu, and T. Chua, “Discrete collaborative filtering.” in *SIGIR*, 2016, pp. 325–334.
- [15] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted boltzmann machines for collaborative filtering.” in *ICDM*, 2007, pp. 791–798.
- [16] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 5, 1989.
- [17] I. Bayer, X. He, B. Kanagal, and S. Rendle, “A generic coordinate descent framework for learning from implicit feedback.” in *WWW*, 2017.
- [18] S. Sedhain, A. Menon, S. Sanner, and L. Xie, “Autorec: Autoencoders meet collaborative filtering.” in *WWW*, 2015, pp. 111–112.
- [19] Y. Zheng, B. Tang, W. Ding, and H. Zhou, “A neural autoregressive approach to collaborative filtering.” in *ICML*, 2016, pp. 764–773.
- [20] A. Elkahky, Y. Song, and X. He, “A multi-view deep learning approach for cross domain user modeling in recommendation systems.” in *WWW*, 2015, pp. 278–288.
- [21] F. Strub and J. Mary, “Collaborative filtering with stacked denoising autoencoders and sparse inputs.” in *NIPS Workshop on Machine Learning for eCommerce*, 2015.
- [22] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data.” in *NIPS*, 2013, pp. 2787–2795.
- [23] T. Cheng, L. Koc, J. Harmsen, and T. Shaked, “Wide and deep learning for recommender systems,” in *WWW*, 2016, pp. 2787–2795.
- [24] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” in *NIPS*, 2008, pp. 1–8.
- [25] C. T. He, X. M. Kan, and X. Chen, “Trirank: Review-aware explainable recommendation by modeling aspects,” in *CIKM*, 2001, pp. 285–295.
- [26] X. Geng, H. Zhang, J. Bian, and T. Chua, “Learning image and user features for recommendation in social networks,” in *ICCV*, 2015, pp. 4274–4282.
- [27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *WWW*, 2015, pp. 1661–1670.
- [28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Item-based collaborative filtering recommendation algorithms,” in *WWW*, 2015, pp. 1661–1670.

- [29] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2014, pp. 1–15.
- [30] X. Wang, L. Nie, X. Song, D. Zhang, and T. Chua, “Unifying virtual and physical worlds: Learning towards local and global consistency,” *ACM Transactions on Information Systems*, 2017.
- [31] X. He, M. Kan, P. Xie, and X. Chen, “Comment-based multi-view clustering of web 2.0 items,” in *WWW*, 2014, pp. 771–781.
- [32] I. Bayer, X. He, B. Kanagal, and S. Rendle, “A generic coordinate descent framework for learning from implicit feedback.” in *WWW*, 2017.