Neural Matrix Factorization Model on Movie-Lens 100K

Sunny Bhati
Indiana University Bloomington
sbhati@iu.edu

Biraj Parikh
Indiana University Bloomington
bparikh@iu.edu

Abstract

We have implemented three algorithms, Generalized Matrix Factorization (GMF), Neural Collaborative Filtering(MLP) and Neural Matrix Factorization Model (NMF). We used non-linear approximators such as Deep neural Networks for all the three algorithms. We also investigated how pre-training the model affects the performance of the model. To assess the performance of the model, we have used Hit Ratio and NDCG as the metrics. We also investigated different architectures, embedding size and regularization coefficient to improve the performance. Due to computational complexity, we have restricted our number of episodes to 100.

1 Introduction

In the era of information explosion, recommender systems play a pivotal role in alleviating information overload, having been widely adopted by many online services, including E-commerce, online news and social media sites. The key to a personalized recommender system is in modelling users' preference on items based on their past interactions (e.g., ratings and clicks), known as collaborative filtering. The popular technique ,among the various collaborative filtering techniques, is matrix factorization (MF) which projects users and items into a shared latent space, using a vector of latent features to represent a user or an item. Thereafter a user's interaction on an item is modelled as the inner product of their latent vectors. The inner product, which simply combines the multiplication of latent features linearly, may not be sufficient to capture the complex structure of user interaction data. In this project, we use **implicit** feedback rather than the user provided ratings and use them to train our non-linear models.

2 Dataset and Features

MovieLens 100k dataset is used for the building the recommendation algorithm, which is available on Kaggle, the dataset consists of 100k number of ratings from over 943 users, and the total number of unique movies is 1682. There are two files namely u.data and u.item. The u.data consists of features like user_id, movie_id, ratings, and timestamp while the u.item consists of movie_id, title, release_date, genre, imdb_url.

3 Background

Let M and N denote the number of users and items, respectively. We define the user–item interaction matrix y_{ui} as 1 if interaction is observed else 0. The recommendation problem with implicit feedback is formulated as the problem of estimating the scores of unobserved entries in Y, which are used for ranking the items. Model-based approaches assume that data can be generated (or described) by an underlying model. We can formulate the problem as $\hat{y}_{ui} = F(u, i \mid 0)$. where $\hat{y}_{ui}$ denotes the predicted score of interaction y_{ui}.

3.1 Neural Collaborative Filtering

Based on the figure Fig, we can see that NCF adopts a multi-layer perceptron framework to model a user-item interaction. The inputs to the model are feature vectors related to user u and item i, which can customize to support range of modelling of user and items. In our case, since the approach is purely formalizing based on collaborative filtering framework, we have used one-hot encoding vector for user u and item i to model the interaction.

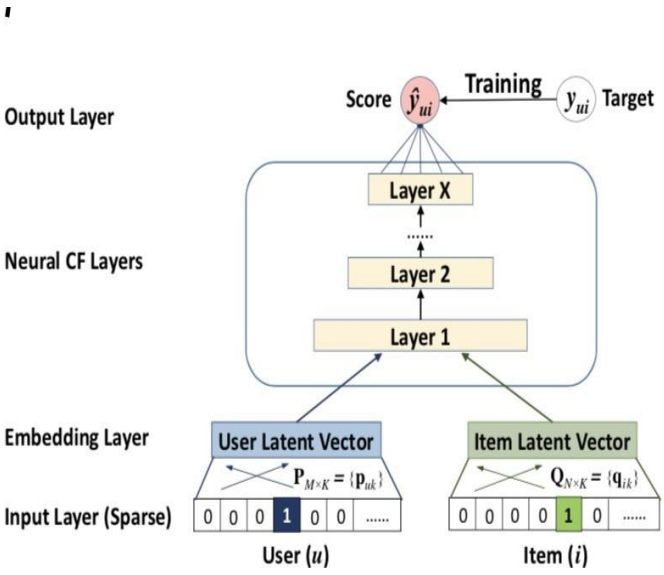


Figure 1: Neural Collaborative Filtering

To introduce the non-linearity in the model, we have used **ReLu** activation function to capture non-linear interaction between user u and item i. Moreover, due to the one-class nature of implicit feedback, Binary Cross Entropy Loss is used as an objective function.

The equation can be seen below:

$$L = -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j)\in\mathcal{Y}^-} \log(1-\hat{y}_{uj})$$
$$= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log \hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}).$$

Figure 2: Binary Cross Entropy Loss

3.2 Generalized Matrix Factorization

Matrix Factorization (MF) is the most widely and popular studied algorithm. Under the Collaborative Framework, MF can be generalized and extended. Due to the one-hot encoding of the user and the item, the embedding vector is the latent vector for user u and item i. The mapping function of the neural layer for GMF can be defined as $\phi_1(p_u, q_u) = p_u \, q_u$. Hadamard product is computed between p_u and q_u. We then project the vector to the output layer followed by a sigmoid activation at the end: $\hat{y_{ui}} = a_{out}(h^T(p_u \cdot q_u))$.

Here, a_{out} refers to the activation function and h refers to the edge weights of the output layer.

3.3 Neural Matrix Factorization Model

Neural Matrix Factorization Model (NMF) is a fusion of GMF and Neural Collaborative Filtering. The following diagram describes the architecture.

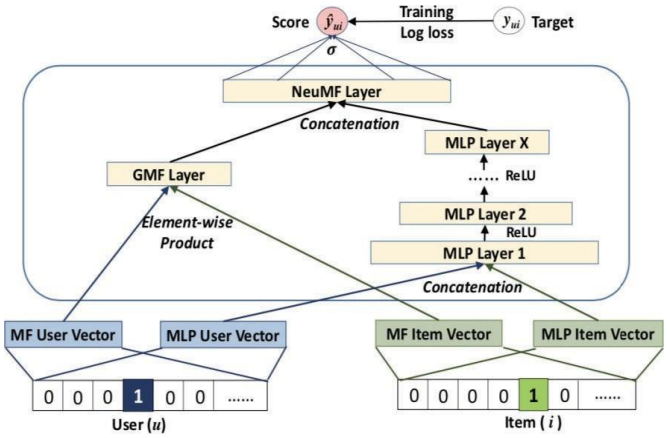


Figure 3: Neural Matrix Factorization Model

Based on the above diagram, we can see that both GMF and NCF process share different embeddings which give flexibility to the model. As discussed before, we used ReLu as the activation function of MLP layers. This model combines the linearity of MF and non-linearity of DNNs for modelling user–item latent structures. The model is called "**NeuMF**", Neural Matrix Factorization.

4. Training and Test Creation

For Training, we have created the training set by selecting all the user movie interactions except the most recent one. To illustrate, for a user u, if the particular user u has rated 10 movies, we have selected the 9 out of 10 interaction as the training examples and kept the most recent interaction for testing. This methodology is followed for every user. In addition, we sample 4 negative samples (by negative samples, selecting movies that have not been rated by user) related to a user for every positive interaction data which has been selected.

5. Evaluation Protocol

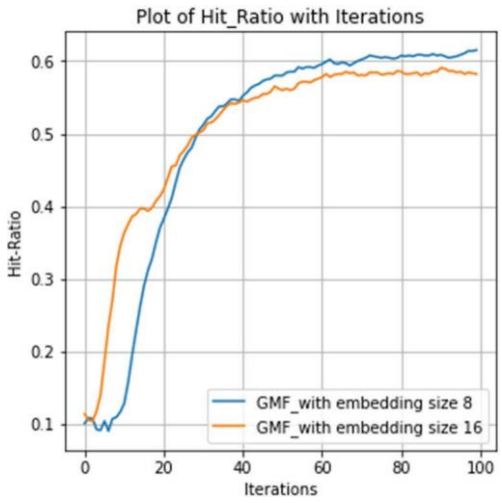
To evaluate the performance of item recommendation, we adopted the **leave-one-out** evaluation, which has been widely used in literature. For each user, we held-out the latest interaction as the test set and utilized the remaining data for training. Since it is too time-consuming to rank all items for every user during evaluation, we followed a strategy that randomly samples 100 items that are not interacted by the user, ranking the test item among the 100 items. The performance of a ranked list is judged by **Hit Ratio (HR)** and **Normalized Discounted Cumulative Gain (NDCG)**. Without special mention, we truncated the ranked list at 10 for both metrics. As such, the HR intuitively measures whether the test item is present on the top-10 list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score.

6. Experiments

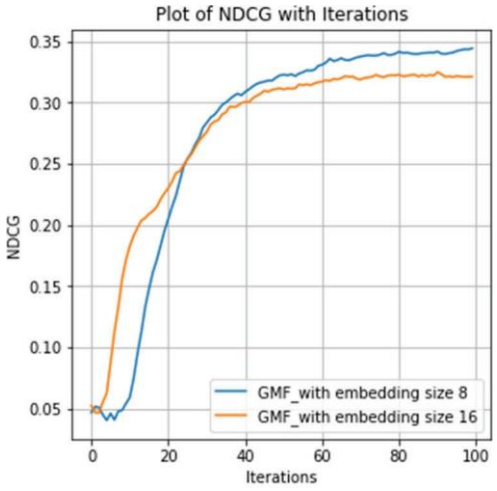
We have conducted various experiments for the three different models using PyTorch framework. We have also used Adam Optimizer with learning rate 0.001 and different regularization coefficient for our experiment. We have discussed in detail the experiments for different models.

6.1 Generalized Matrix Factorization

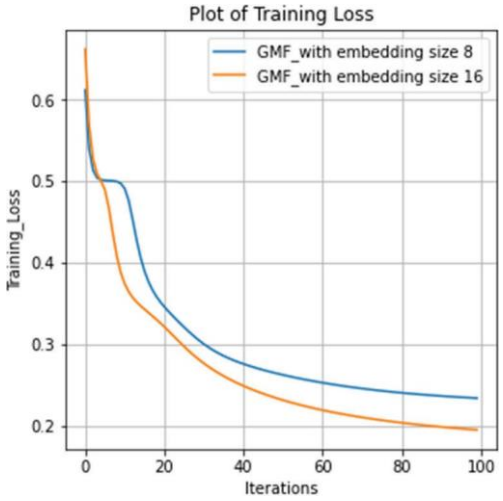
We evaluated the GMF for embedding size 8 and 16 . Below are the graphs of NDCG and Hit Ratio with the iterations. We used a batch size of 256 for creating the training set for the GMF model.



(a) Plot of Hit-Ratio



(b) Plot of NDCG



(c) Plot of Training Loss

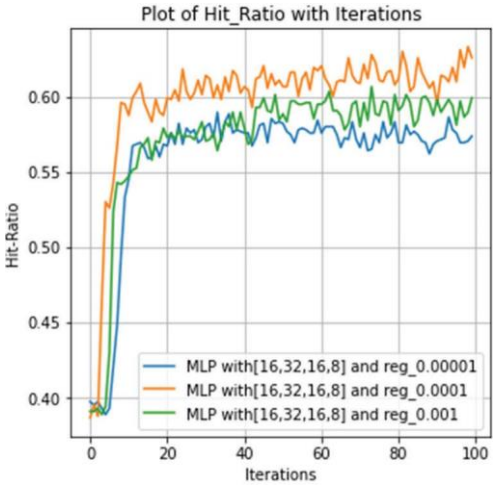
Figure 4: Generalized Matrix Factorization

Based on the graph , we can observe that Generalized Matrix Factorization works well with latent dimension size of 8.

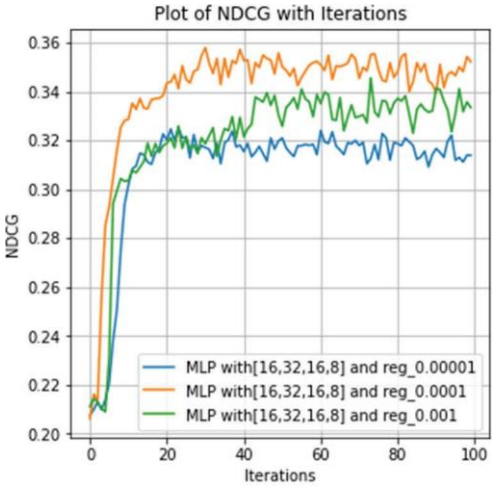
6.2 Neural Collaborative Filtering

For Neural Collaborative Filtering, we have used a batch size of 1024 for creating the training dataset. We also investigated the performance of the NCF for two different architectures [16,32,16,8] and [16,64,32,16,8] and used ReLu activation function for the hidden layers, followed by sigmoid activation at the output layer. Both the architectures mentioned above were tested with different regularization coefficients of 0.001, 0.0001 and 0.00001 for finding the optimal efficiency.

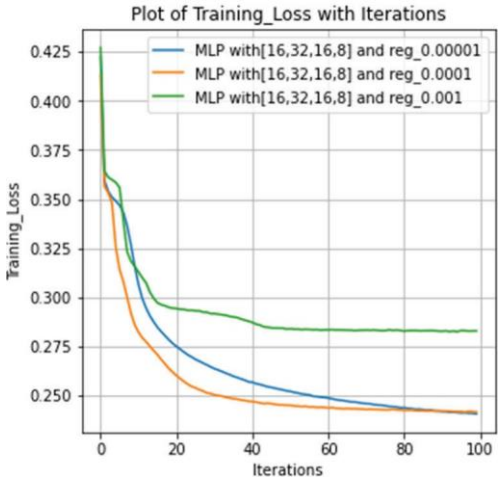
It can be observed that from the above plots with regularization coefficient 0.0001 for the architecture [16,32,16,8], the performance of the model is superior for both the metrics NDCG and Hit Ratio.



(a) Plot of Hit-Ratio

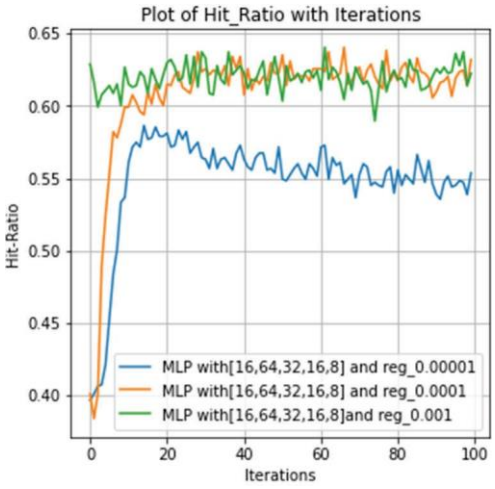


(b) Plot of NDCG

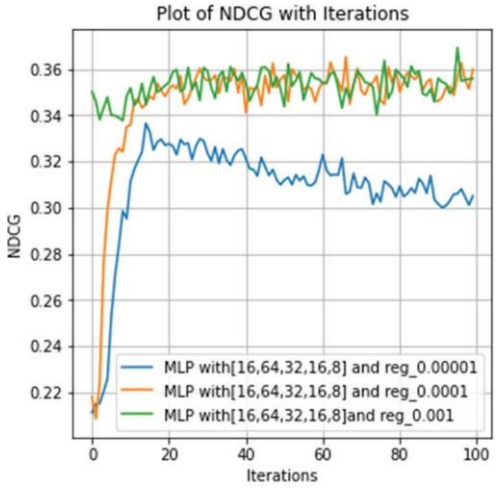


(c) Plot of Training Loss

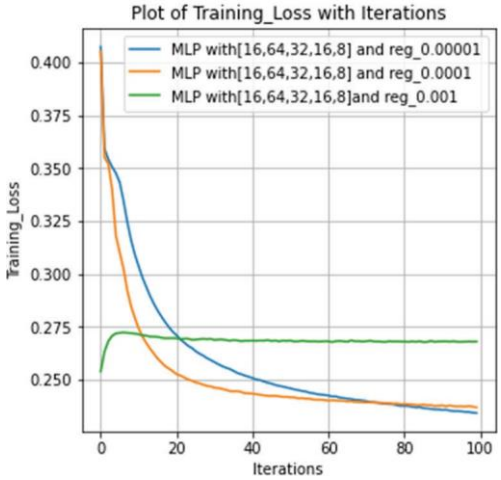
Figure 5: Neural Collaborative Filtering for architecture [16,32,16,8]



(a) Plot of Hit-Ratio



(b) Plot of NDCG

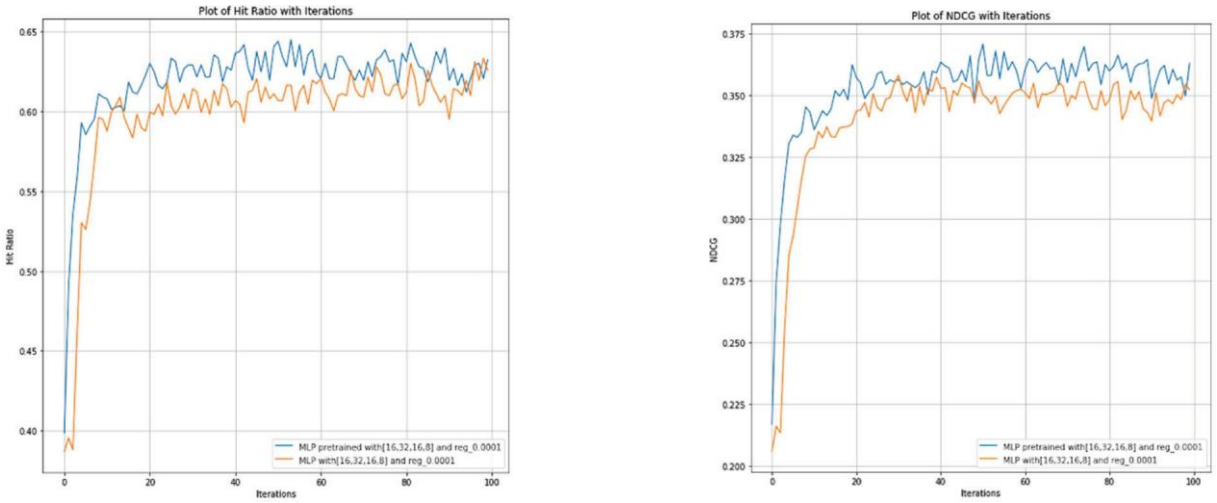


(c) Plot of Training Loss

Figure 6: Neural Collaborative Filtering for architecture [16,64,32,16,8]

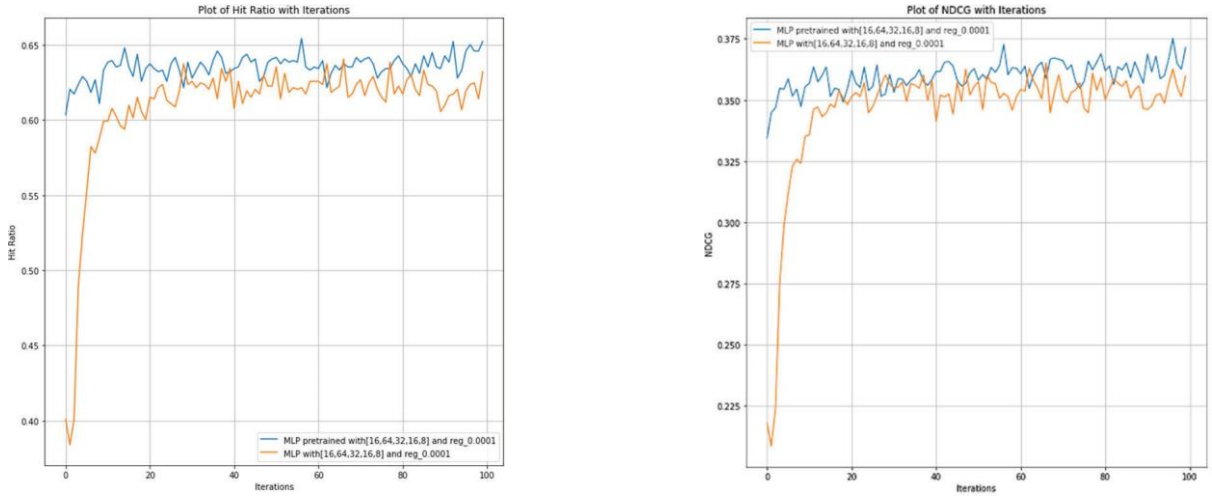
From the above plots, we can observe that the model performance for the architecture [16,64,32,16,8] with the regularization coefficient 0.0001 is better for the both the metrics Hit-Ratio and NDCG. It is evident from the plots for both the architecture, we can observe that the performance of the model for regularization coefficient 0.0001 is better.

We further experimented and initialized the weights of the embedding layer of NCF by the pre-trained weights obtained from Generalized Matrix Factorization for both the architecture with regularization coefficient 0.0001. Below are the plots of the results obtained by initializing the NCF model with pre-trained weights vs randomly initialized weights for NCF model:



(a) Plot of Hit-Ratio (b) Plot of NDCG

Figure 7: Comparison of Pretrained vs No-Pretraining Neural Collaborative Filtering for architecture [16,32,16,8]



(a) Plot of Hit-Ratio **(b) Plot of NDCG**

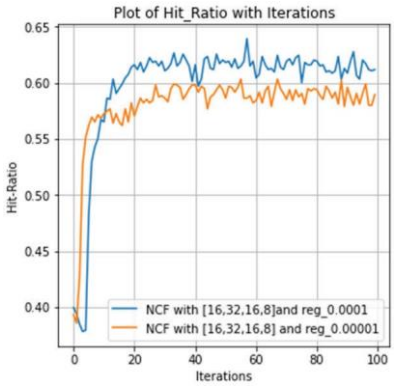
Figure 8: Comparison of Pretrained vs No-Pretraining Neural Collaborative Filtering for architecture [16,32,16,8] and [16,64,32,16,8]

It is evident that based on the graphs above pre training boosts the performance of the model.

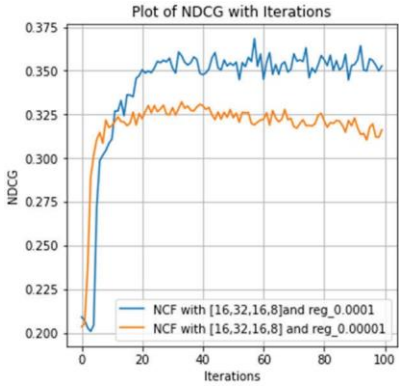
6.3 Neural Matrix Factorization Model

For Neural Matrix Factorization Model, we have used a batch size of 256 for training the data. We also investigated the performance of the NCF for two different architectures [16,32,16,8] and [16,64,32,16,8]. We have separate embedding for GMF and NCF with dimension size 8.We used ReLu activation function for the hidden layers, followed by sigmoid activation at the output layer. We investigated both the architecture mentioned above with regularization coefficients of 0.0001 and 0.00001. Below are the plots of the model trained with random initialization.

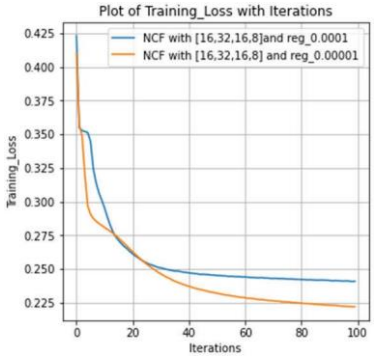
Below are the plots for architecture [16,32,16,8] and [16,64,32,16,8] with regularization coefficients of 0.0001 and 0.00001.



(a) Plot of Hit-Ratio

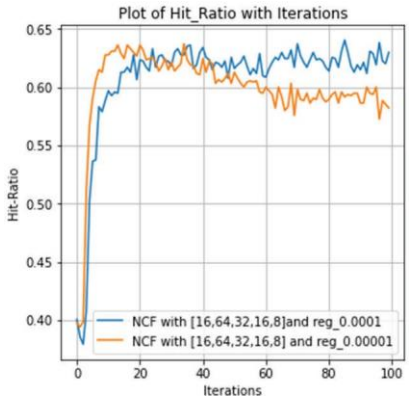


(b) Plot of NDCG

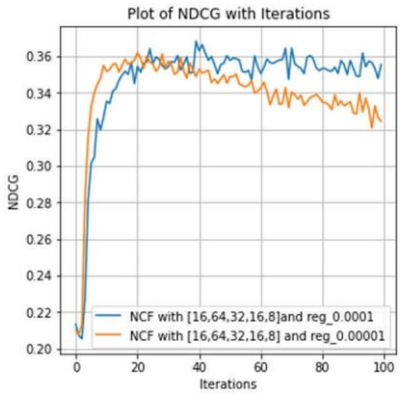


(c) Plot of Training Loss

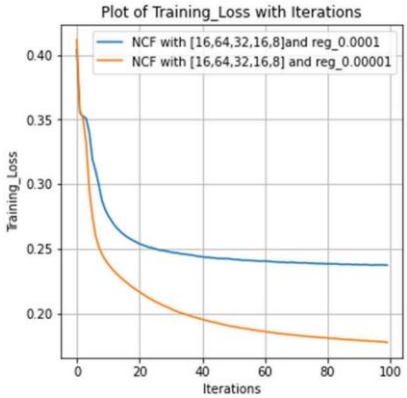
Figure 9: Neural Matrix factorization for architecture [16,32,16,8]



(a) Plot of Hit-Ratio



(b) Plot of NDCG

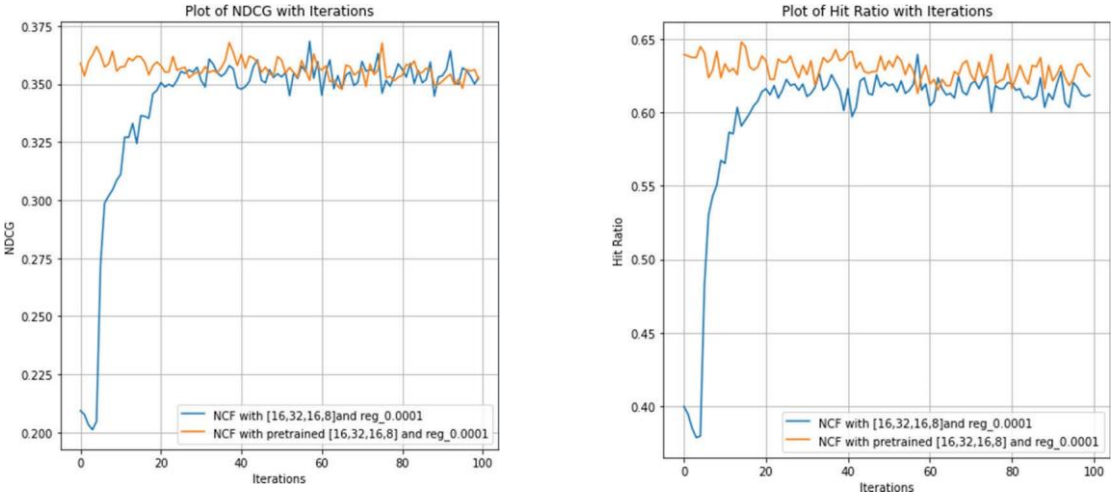


(c) Plot of Training Loss

Figure 10: Neural Matrix factorization for architecture [16,64,32,16,8]

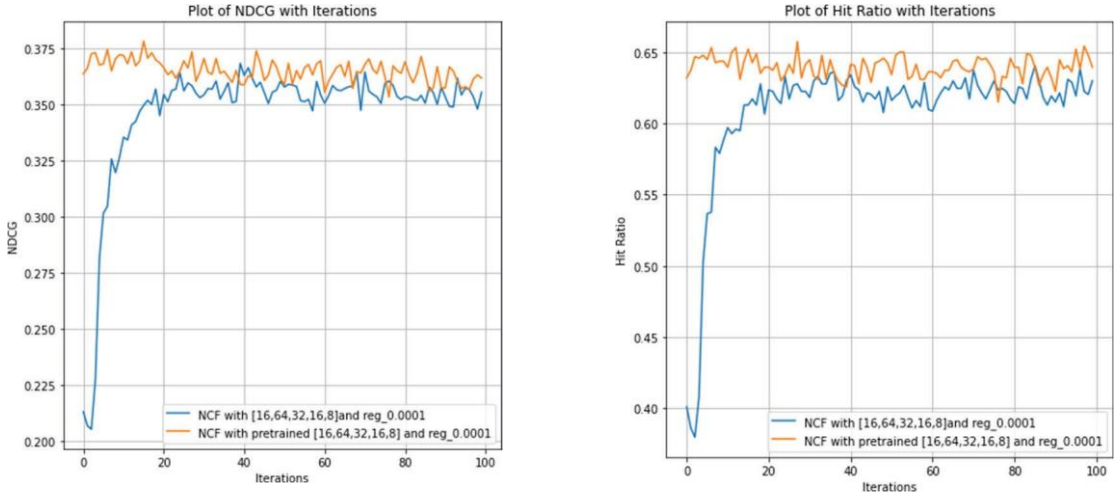
It can be observed that from the above plots for both the architecture with regularization coefficient 0.0001, the performance of the model is superior for both the metrics NDCG and Hit Ratio.

Further, we performed a comparison between the NMF trained from random initialization vs the NMF trained by using the pretrained weights from NCF and GMF model. Below are the plots for both the architecture with regularization coefficient 0.0001.



(a) Plot of NDCG **(b) Plot of Hit-Ratio**

Figure 11: Comparison of Pretrained vs Random Initialization Neural Matrix factorization for architecture [16,32,16,8]



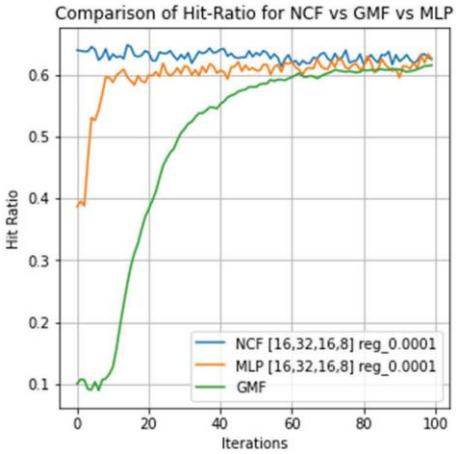
(a) Plot of NDCG **(b) Plot of Hit-Ratio**

Figure 12: Comparison of Pretrained vs Random Initialization Neural Matrix factorization for architecture [16,64,32,16,8]

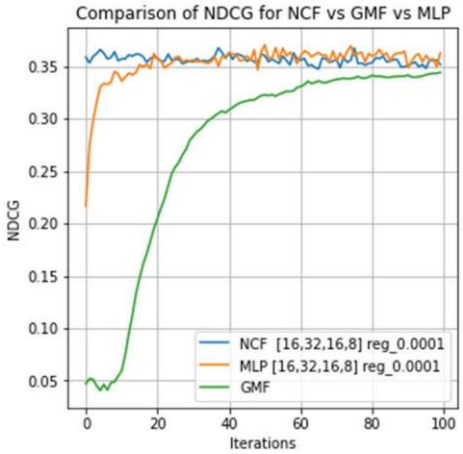
Based on the above figure, we can see that the due to pretraining the starting metric value ,i.e Hit Ratio and NDCG starts quite high and converges faster.

7 Final Results: Comparison of NMF vs NCF vs GMF

Finally, we compared the evaluation of the performance of the model for both the architectures [16,32,16,8] and [16,64,32,16,8].We have selected 0.0001 as the ideal regularization coefficient based on our analysis as show in the previous section. Below are the plots comparing NMF ,NCF and GMF.

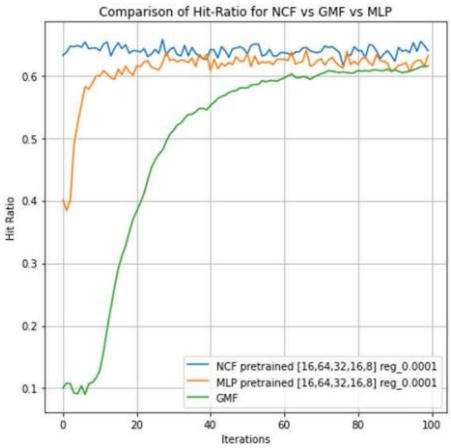


(a) Plot of Hit Ratio

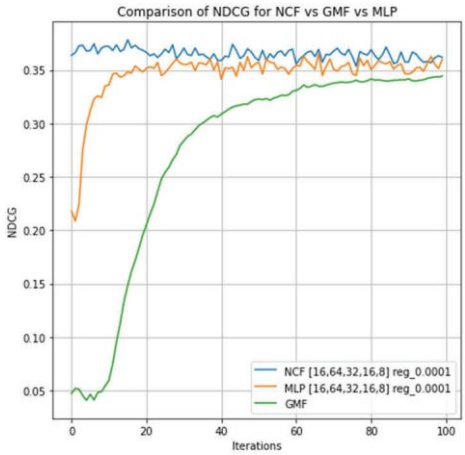


(b) Plot of NCDCG

Figure 13: Comparison of NMF vs NCF vs GMF for architecture [16,32,16,8]



(a) Plot of Hit Ratio



(b) Plot of NCDCG

Figure 14: Comparison of NMF vs NCF vs GMF for architecture [16,64,32,16,8]

Based on the graph, due to the flexibility that NMF model provides, we can conclude that NMF outperforms NCF and NCF outperforms GMF. First, we can see that with more iterations, the training loss of NCF models gradually decreases and the recommendation performance is improved. The most effective updates are occurred in the first 10 iterations, and more iterations may overfit a model. Second, among the three NCF methods, NeuMF shows the better performance with regard to the metrics observed, followed by MLP, and then GMF. The above finding provides empirical evidence for the rationality and effectiveness of optimizing the log loss for learning from implicit data.

8 Reference

1) https://arxiv.org/abs/1708.05031
2) http://sumanthrb.com/ml/recommendation-techniques/
3) https://www.ijcai.org/Proceedings/2018/0308.pdf