PAPER REVIEW (22-04-12)

Neural Collaborative Filtering

Xiangnan He, Lizi Liao, Hanwang Zhang,
Liqiang Nie, Xia Hu, Tat-seng Chua

WWW '17: Proceedings of the 26th International Conference on World Wide Web, 2017.

서울시립대학교
UNIVERSITY OF SEOUL

컴퓨터과학과
G202292003 이희태

CONTENTS

BASICS OF RECOMMENDATION SYSTEM

What do they have in common? Recommendation system!

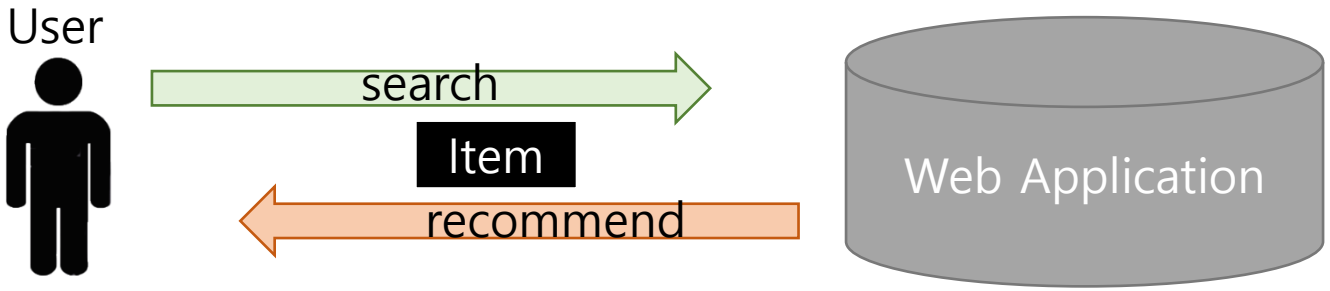
BASICS OF RECOMMENDATION SYSTEM

Why use recommendation systems?

"The Long Tail"

BASICS OF RECOMMENDATION SYSTEM

Search engine vs. Recommender engine



Search engine	Recommendation engine
Pull information	Push information
사용자가 요구한 후 작동	사용자가 요구하기 전에 작동
사용자 본인이 원하는 바를 알고 있음	사용자 본인이 원하는 게 무엇인지 정확히 알고 있지 못함

BASICS OF RECOMMENDATION SYSTEM

Terminology

- Users
  - Clients of the platform.

- Items
  - Products list to recommend of the platform.
  - books, films, news articles, SNS post, music, games, etc.

- Interactions
  - Explicit Feedback (e.g., ratings, Like-dislike)
  - Implicit Feedback (e.g., clicking patterns, buying history, time spent on the site, search log)
  - Can be represented by an utility matrix.

- Profiles(Features)
  - The vectors describing users or items.
  - It is essential to select profiles that represents the characteristics of the items well.

BASICS OF RECOMMENDATION SYSTEM

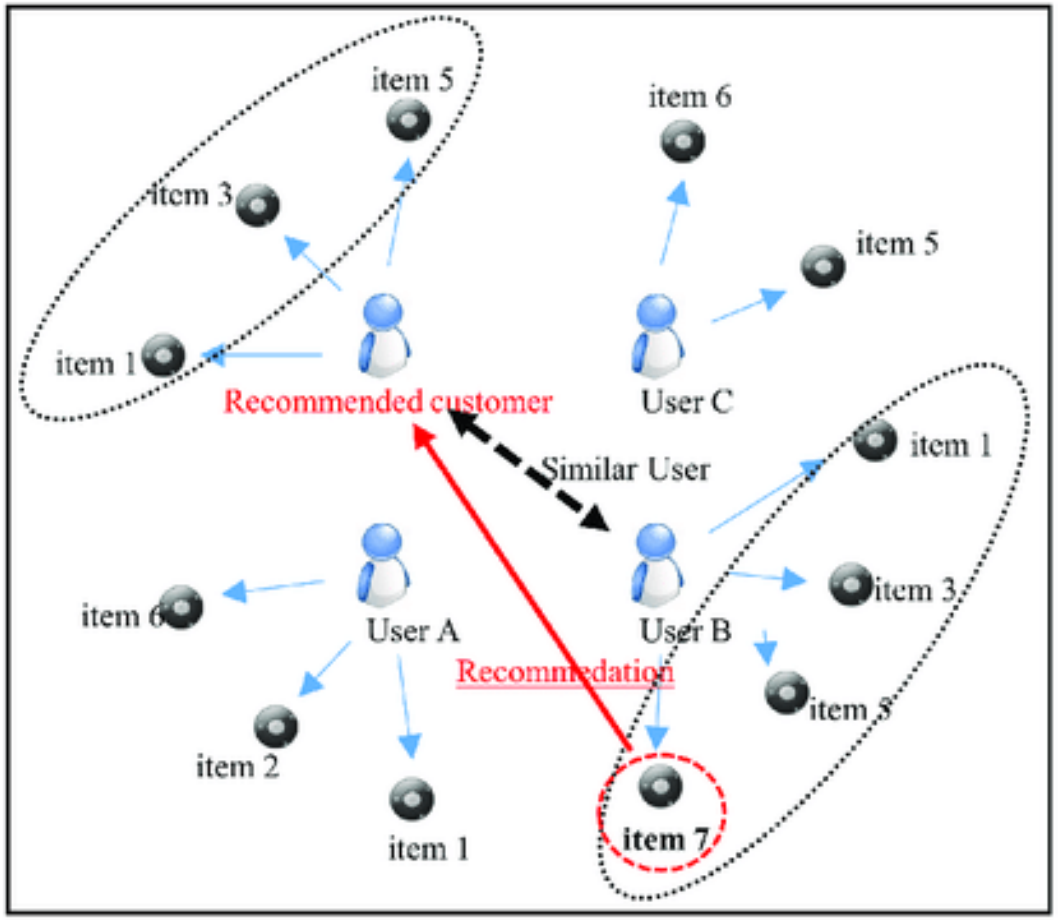
Problems that the recommendation system can solve

1. Ranking
    - 특정 item을 좋아할 만한 top-K명의 user들을 선정하는 문제
    - 특정 user가 좋아할 만한 top-K개의 item들을 선정하는 문제

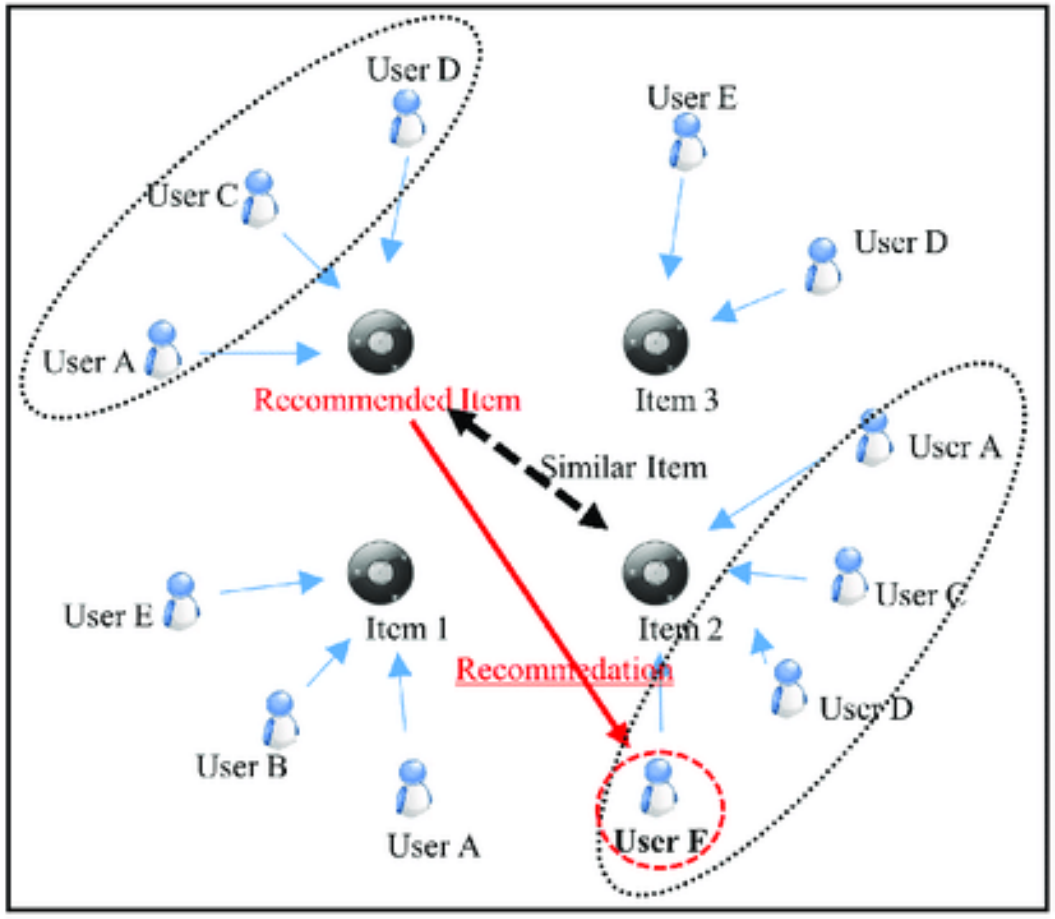
2. Prediction
    - User-Item interaction matrix의 missing values를 채우는 문제
    - User-Item interaction matrix is a sparse matrix.
        - Observed value (관측값)
        - Non-observed value (=missing value, 결측값)

BASICS OF RECOMMENDATION SYSTEM

Basic Concept of Collaborative Filtering



(A)



(B)

Basic Concept of Collaborative Filtering (cont'd)

Movies

Users	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

BASICS OF RECOMMENDATION SYSTEM

Challenges In Collaborative Filtering

1. Cold start
   - 충분한 수의 user들이 확보되지 못하면(cold), 서비스가 원활히 작동하지 않는다.

2. Sparsity
   - 많은 수의 결측값 요소로 인해 같은 items list에 대해 평가를 내린 다른 user를 찾기가 어렵다.

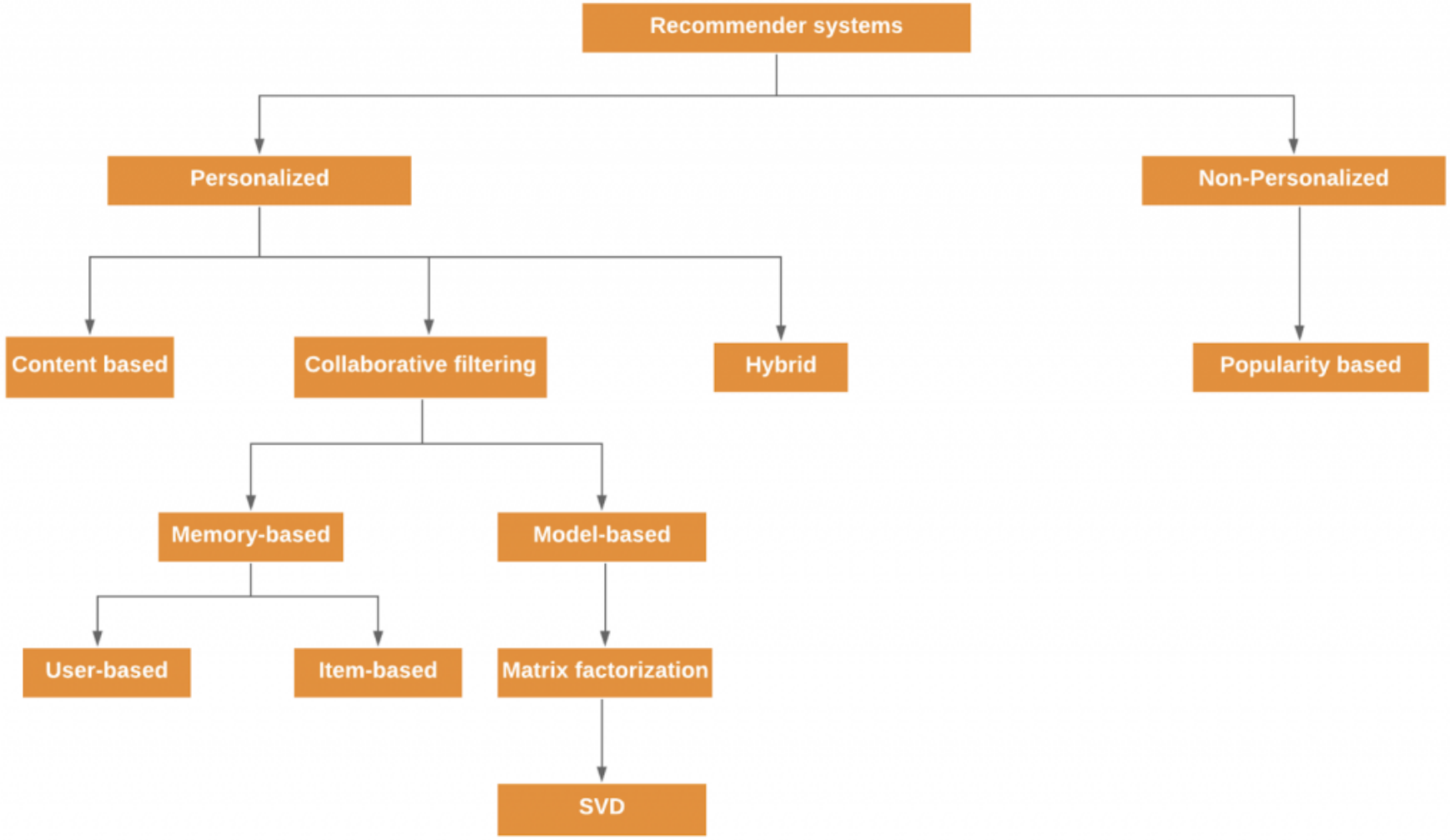
3. First rater
   - 실제 서비스에서는 매 시각마다 새로운 아이템이 꾸준히 생성된다.
   - 어느 누구에 의해서도 평가를 받지 못한 item은 user들에게 추천하기 어렵다.

4. Popularity bias
   - 특이한 취향을 가진 사람에게 추천을 하기가 힘들다.
   - 소수 장르보다는 대중적인 item들을 추천하려는 경향이 강하다.

5. Security
   - Implicit feedback 기반일 경우 기업이 사용자 개인정보를 악용/남용할 가능성이 존재한다.

BASICS OF RECOMMENDATION SYSTEM

Types of Recommendation System



ABSTRACT

In recent years, deep neural networks have yielded immense success on speech recognition, computer vision, and NLP, but relatively little research has been done on recommendation systems.

In this work, **we strive to develop techniques based on neural networks to tackle the key problem in recommendation - collaborative filtering - on the basis of implicit feedback.**

Although some recent work has employed deep learning for recommendation, they primarily used it to model auxiliary information.

When it comes to model the key factor in collaborative filtering - the interaction between user and item features, they still resorted to matrix factorization and applied an inner product on the latent features of users and items.

ABSTRACT

By replacing the inner product with a neural architecture, we present a general framework named **NCF**, short for ***Neural networkbased Collaborative Filtering***.

NCF is generic and can express and generalize matrix factorization under its framework.

To supercharge NCF modelling with non-linearities, we propose to leverage a multi-layer perceptron to learn the user–item interaction function.

Extensive experiments on two real-world datasets show significant improvements of our proposed NCF framework over the state-of-the-art methods.

Extensive experiments on two real-world datasets show significant improvements of our proposed NCF framework over the state-of-the-art methods.

01. INTRODUCTION

01 Introduction

- In the era of information explosion, recommender systems play a pivotal role in alleviating information overload.
- 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*** **(CF)**.
- Among the various collaborative filtering techniques, **matrix factorization (MF)** is the most popular one.
- Popularized by the Netflix Prize, MF has become the *de facto* approach to latent factor model-based recommendation.
- Much research effort has been devoted to enhancing MF.
  - Integrating it with neighbor-based models
  - Combining it with topic models of item content
  - Extending it to factorization machines for a generic modelling of features.

01 Introduction

- Despite the effectiveness of MF for collaborative filtering, it is well-known that its performance can be hindered by the simple choice of the interaction function - inner product.
- This paper explores the use of deep neural networks for learning the interaction function from data.
- Recently deep neural networks (DNNs) have been found to be effective in several domains, ranging from computer vision, speech recognition, to text processing.
- However, there is relatively little work on employing DNNs for recommendation in contrast to the vast amount of literature on MF methods.
- Some recent studies mostly used DNNs to model auxiliary information, such as textual description of items, audio features of music, and visual content of images.
- With regards to modelling the key collaborative filtering effect, they still resorted to MF, combining user and item latent features using an inner product.

01 Introduction

- This work addresses the aforementioned research problems by formalizing a neural network modelling approach for collaborative filtering. We focus on **implicit feedback**.
- Compared to explicit feedback, implicit feedback can be tracked automatically and is thus much easier to collect for content providers.
- However, it is more challenging to utilize, since user satisfaction is not observed and there is a natural scarcity of negative feedback.
- **In this paper, we explore the central theme of how to utilize DNNs to model noisy implicit feedback signals.**

01 Introduction

The main contributions of this work are as follows.

1. We present a neural network architecture to model latent features of users and items and devise a general framework NCF for collaborative filtering based on neural networks.

2. We show that MF can be interpreted as a specialization of NCF and utilize a multi-layer perceptron to endow NCF modelling with a high level of non-linearities.

3. We perform extensive experiments on two real-world datasets to demonstrate the effectiveness of our NCF approaches and the promise of deep learning for collaborative filtering.

02. PRELIMINARIES

2.1 Learning from Implicit Data
2.2 Marix Factorization

User – Item interaction matrix

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	0	0	1	0
User 2	0	0	0	0	1
User 3	0	0	0	1	0
User 4	0	0	1	0	0

Observed Interaction

Unobserved Interaction

We define the user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ from users' implicit feedback as,

$$y_{ui} = \begin{cases} 1, & \text{if interaction (user } u\text{, item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

- 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.

- Formally, they can be abstracted as learning

$$\hat{y}_{ui} = f(u, i|\theta),$$

where $\hat{y}_{ui}$ denotes the predicted score of interaction y_{ui}

- To estimate parameters θ, existing approaches generally follow the machine learning paradigm that optimizes an objective function.

- Two types of objective functions are most commonly used in literature.
  1. **Pointwise loss**
  2. **Pairwise loss**

- Our NCF framework supports both pointwise and pairwise learning.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0.7	0	0	1.1	0
User 2	0	0	0.5	0	0.4
User 3	0.5	0	0	0.7	0
User 4	0	0	0.6	0	0.5

Calculate error

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	0	0	1	0
User 2	0	0	0	0	1
User 3	0	0	0	1	0
User 4	0	0	1	0	0

Utility matrix

≈

0	1.3
-0.6	0
0	0.9
-0.8	0

User matrix

x

0	0	-0.8	0	-0.6
0.5	0	0	0.8	0

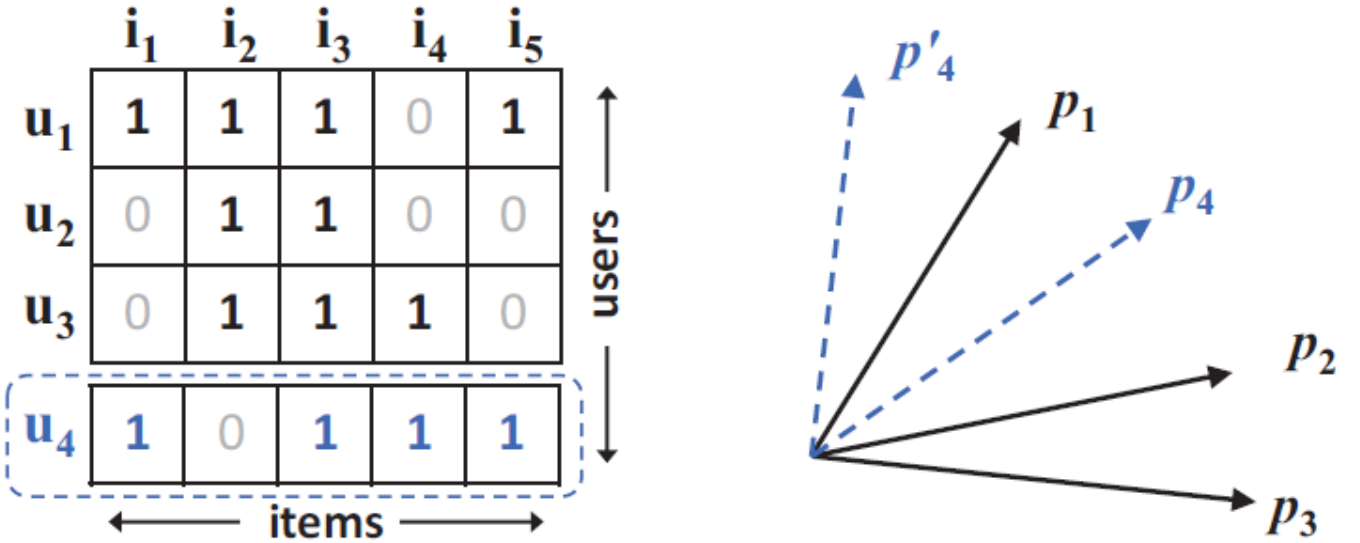
Item matrix

- MF associates each user and item with a real-valued vector of latent features.
- Let P_u and Q_i denote the latent vector for user *u* and item *i*, respectively; MF estimates an interaction $\hat{y}_{ui}$ as the inner product of P_u and Q_i :

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

  where K denotes the dimension of the latent space.

- As such, MF can be deemed as a linear model of latent factors.



(a) user–item matrix (b) user latent space

Figure 1: An example illustrates MF's limitation.

- This example shows the possible limitation of MF caused by the use of a simple and fixed inner product to estimate complex user-item interactions in the low-dimensional latent space.
- In this work, we address the limitation by learning the interaction function using DNNs from data.

03. NEURAL COLLABORATIVE FILTERING

3.1 General Framework
3.2 Generalized Matrix Factorization(GMF)
3.3 Multi-Layer Perceptron(MLP)
3.4 Fusion of GMP and MLP

General Framework

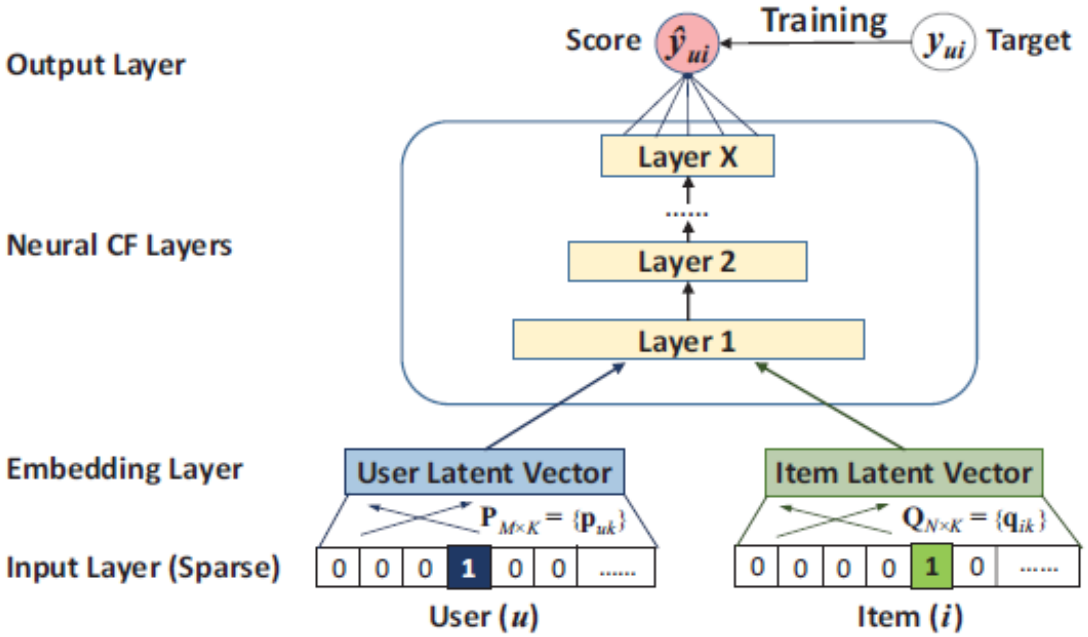


Figure 2: Neural collaborative filtering framework

- Input Layer: one-hot encoding vector
- Embedding Layer: This projects the sparse representation to a dense vector (in the latent space)
- User/Item latent vectors are fed into a multi-layer neural architecture, which we term as *neural collaborative filtering layers*.
- The final output layer produced score $\hat{y}_{ui}$, and training is performed by minimizing the loss between $\hat{y}_{ui}$ and its target value $\hat{y}$.

Formulas of the NCF's predictive model

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

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

P

a	b	c
d	e	f
g	h	i
j	k	l

$\mathbf{v}_u^U$

0
0
1
0

$\mathbf{P}^T\mathbf{v}_u^U$ =

a	d	g	j
b	E	h	k
c	F	i	l

x

0
0
1
0

$\mathbf{P} \in \mathbb{R}^{M \times K}$: the latent factor matrix for users. $\mathbf{Q} \in \mathbb{R}^{N \times K}$: the latent factor matrix for items.
K : the dimension of the latent space.
Θ_f : the model parameters of the interaction function *f*.
Φ_{out} : the mapping function for the output layer.
Φ_x : the mapping function for the *x*-th layer.

Learning NCF

- Existing pointwise methods largely perform a regression with squared loss:

$$L_{sqr} = \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2, \qquad (5)$$

Y : the set of observed interactions in **Y**
Y^- : the set of negative instances, which can be all (or sampled from) unobserved interactions
W_{ui} : a hyper parameter denoting the weight of training instance (u, i)

- **This loss may not tally well with implicit data!**
- This is because for implicit data, the target value $\hat{y}_{ui}$ is a binarized 1 or 0. (the squared loss can be explained by assuming that observations are generated from a Gaussian distribution)
- The prediction score $\hat{y}_{ui}$ then represents how likely i is relevant to u
- To endow NCF with such a probabilistic explanation, we need to constrain the output in the range of [0, 1]

Learning NCF (cont'd)

- The condition can be easily achieved by using a probabilistic function (*e.g.,* the *Logistic function*) as the activation function for the output layer.

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

- Taking the negative logarithm of the likelihood, we reach

$$\begin{aligned} L &= - \sum_{(u,i) \in \mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j) \in \mathcal{Y}^-} \log(1 - \hat{y}_{uj}) \\ &= - \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}). \end{aligned} \quad (7)$$

Binary cross-entropy loss (a.k.a. *Log loss*)

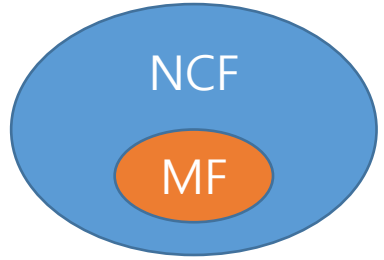
- This is the objective function to minimize for the NCF methods, and its optimization can be done by performing SGD.

- By employing a probabilistic treatment for NCF, we address recommendation with implicit feedback as a binary classification problem.

Generalized Matrix Factorization

3.2 Generalized Matrix Factorization (GMF)

We now show how MF can be interpreted as a special case of our NCF framework.



$$\phi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i, \quad (8)$$

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\phi_1(\mathbf{p}_u, \mathbf{q}_i))), \quad (9)$$

⊙ : element-wise product of vectors
h : edge weights of the output layer
$\boldsymbol{a}_{out}$: activation function
$\boldsymbol{p}_u$: = $\mathbf{P}^T\mathbf{v}_u^U$, the user latent vector , $\boldsymbol{q}_i$: = $\mathbf{Q}^T\mathbf{v}_i^I$, the item latent vector.

In the MF,

$\boldsymbol{a}_{out}$ = identity function, $\mathbf{h}^T$ = $[1, \ldots, 1]_{1xk}$ → dot product of two latent vectors.

Under the NCF framework, MF can be easily generalized and extended.

$\boldsymbol{a}_{out}$ = $\sigma(x) = 1/(1+e^{-x})$, $\mathbf{h}^T$ = $[h_1, \ldots, h_k]$

We term it as **GMF**, short for *Generalized Matrix Factorization*.

Multi-Layer Perceptron

3.3 Multi-Layer Perceptron (MLP)

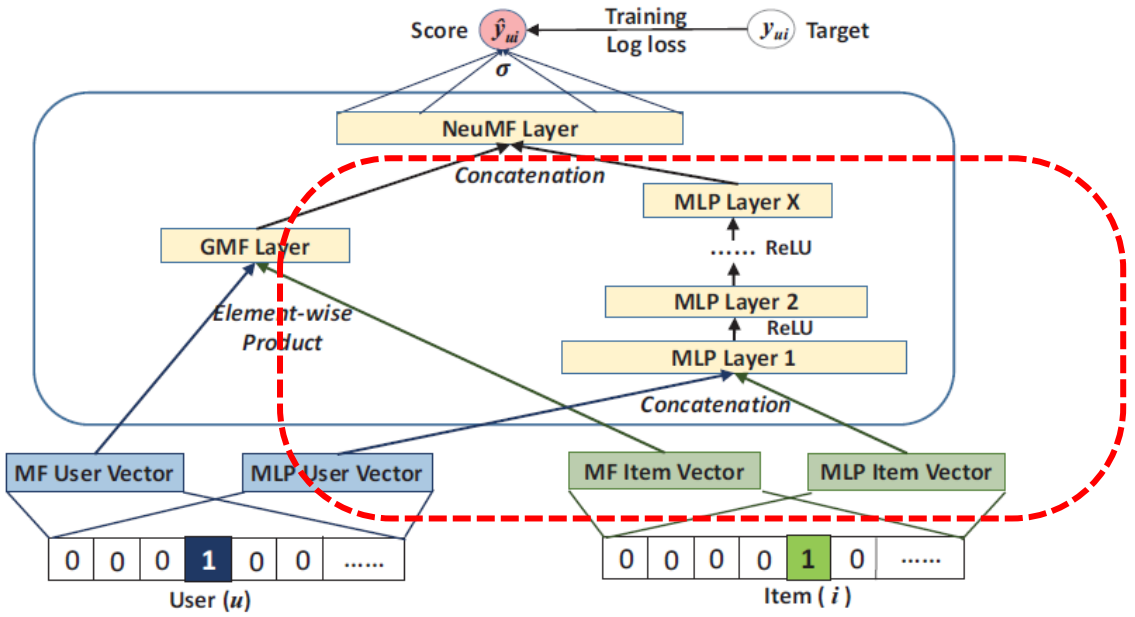


Figure 3: Neural matrix factorization model

- Since NCF adopts two pathways to model users and items, it is intuitive to combine the features of two pathways by concatenating them.
- However, simply a vector concatenation does not account for any interactions between user and item latent features, which is insufficient for modelling the collaborative filtering effect.
- Handling this issue, we propose to using a standard Multi-Layer Perceptron model to learn the interaction between user and item latent features.

3.3 Multi-Layer Perceptron (MLP)

- GMF is difficult to represent the complex relationship between user and item latent features because it is linear and uses only a fixed element-wise product.
- In contrast, MLP is nonlinear and flexible, allowing it to represent more complicated relationships of them.
- The MLP model under our NCF framework is defined as

$$
\begin{aligned}
\mathbf{z}_1 &= \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\
\phi_2(\mathbf{z}_1) &= a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
&\ldots\ldots \\
\phi_L(\mathbf{z}_{L-1}) &= a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L), \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),
\end{aligned}
\tag{10}
$$

Φ_1 : concatenation function of $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$
$\Phi_2 \sim \Phi_L$: Affine functions
$\boldsymbol{a}$: activation functions

We can freely choose sigmoid, hyperbolic tangent, ReLU, etc. as activation functions of MLP layers.

Fusion of GMF and MLP

3.4 Fusion of GMF and MLP

- How can we fuse GMF and MLP under the NCF framework, so that they can mutually reinforce each other to better model the complex user-iterm interactions?
- A straightforward solution is to let GMF and MLP share the same embedding layer.
- However, sharing embeddings of GMF and MLP might limit the performance of the fused model.
- We allow GMF and MLP to learn separate embeddings, and combine the two models by concatenating their last hidden layer.
- ***Neural Matrix Factorization*** (NeuMF)

$\mathbf{p}_u^G$: user embedding for GMF
$\mathbf{p}_u^M$: user embedding for MLP
a_L : activation function of MLP (ReLU)

$$
\begin{aligned}
\phi^{GMF} &= \mathbf{p}_u^G \odot \mathbf{q}_i^G, \\
\phi^{MLP} &= a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L), \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),
\end{aligned}
\tag{12}
$$

3.4 Fusion of GMF and MLP

Neural Matrix Factorization (NeuMF)

= GMF + MLP

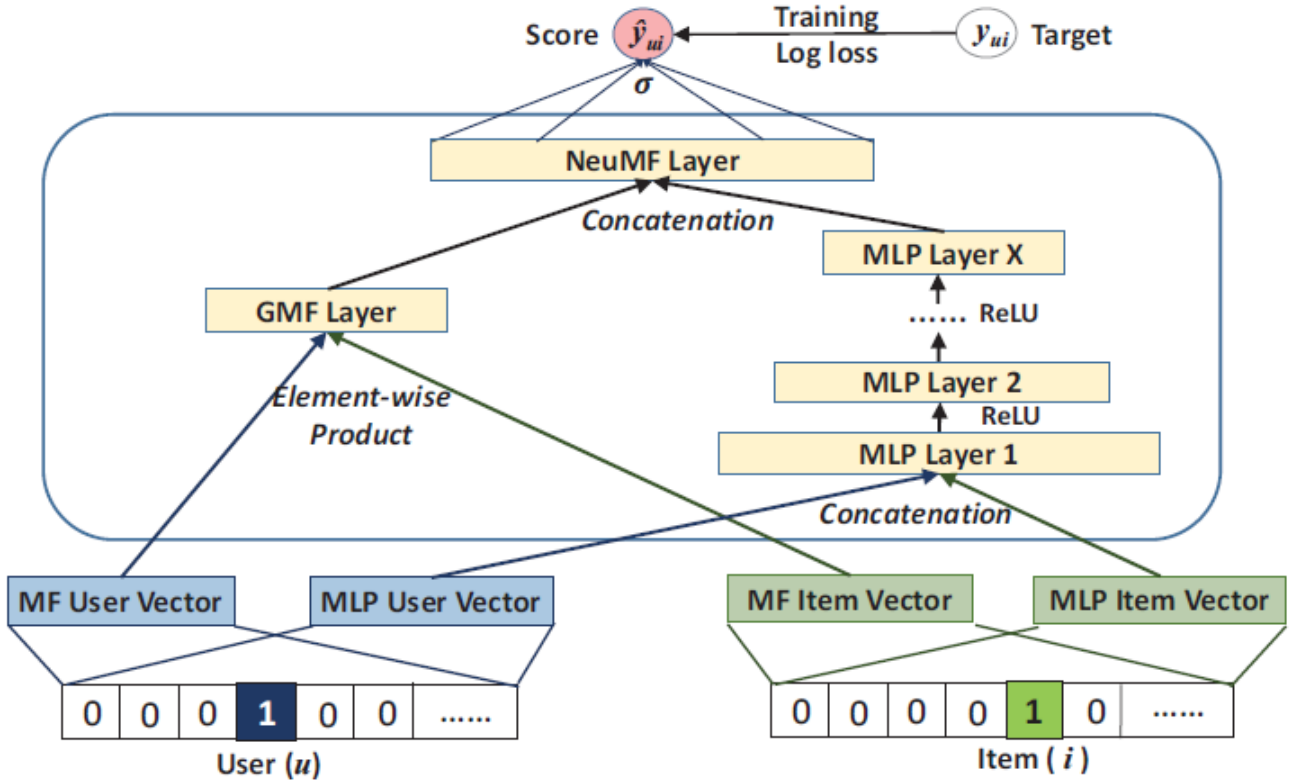


Figure 3: Neural matrix factorization model

3.4 Fusion of GMF and MLP

Pre-training

It is reported that the initialization plays an important role for the convergence and performance of deep learning models. we propose to initialize NeuMF using the pre-trained models of GMF and MLP.

1. First train GMF and MLP with random initializations until convergence.
2. Then use their model parameters as the initialization for the corresponding parts of NeuMF's parameters.
3. Lastly, concatenate with weights of the two models on the output layer.

$$\mathbf{h} \leftarrow \begin{bmatrix} \alpha \mathbf{h}^{GMF} \\ (1-\alpha)\mathbf{h}^{MLP} \end{bmatrix}, \qquad (13)$$

α is a trade-off weight.

- For training GMF and MLP, we adopt the Adam.
- After feeding pre-trained parameters into NeuMF, we optimize it with the vanilla SGD, rather than Adam.
- This is because Adam needs to save momentum information for updating parameters properly.
- Since we have already pre-trained using momentum-based methods, it is unnecessary to use Adam when optimizing NeuMF.

04. EXPERIMENTS

4.1 Experimental Settings
4.2 Performance Comparison (RQ1)
4.3 Log Loss with Negative Sampling (RQ2)
4.4 Is Deep Learning Helpful? (RQ3)

Datasets

Table 1: Statistics of the evaluation datasets.

Dataset	Interaction#	Item#	User#	Sparsity
MovieLens	1,000,209	3,706	6,040	95.53%
Pinterest	1,500,809	9,916	55,187	99.73%

Experimented with two datasets: MovieLens and Pinterest

1. MovieLens (http://grouplens.org/datasets/movielens/1m/)
   - We used the version containing one million ratings, where each user has at least 20 ratings.
   - We transformed explicit feedback data of MovieLens into implicit data.

2. Pinterest (https://sites.google.com/site/xueatalphabeta/academic-projects)
   - This implicit feedback data is constructed by [*X. Geng et al., "Learning image and user features for recommendation in social networks" in ICCV (2015)].*
   - Since high sparseness of dataset, we filtered that in the same way as the MovieLens data that retained only users with at least 20 interactions.

Evaluation Protocols

- We adopted the ***leave-one-out*** evaluation.
- We used user's latest interaction as the test set and utilized the remaining data for training.
- To save time, we followed the common strategy that randomly samples 100 items that are not interacted by user, ranking the test item among the 100 items.
- Two Performance Metrics
  - **Hit Ratio** (HR)
  - **Normalized Discounted Cumulative Gain** (NDCG)
- We truncated the ranked list at 10 for both metrics.
- We calculated both metrics for each test user and reported the average score.

Baselines

- **ItemPop**
  - Ranked by their popularity judged by the number of interactions (Non-personalized method)
- **ItemKNN**
  - A standard item-based CF method
- **BPR**
  - A MF model which is learnt by optimizing a pairwise ranking loss
  - Highly competitive baseline
- **eALS**
  - A state-of-the art MF method using the squared loss (pointwise loss)
  - Treating all unobserved interactions as negative instances and weighting them non-uniformly by the item popularity

Parameter Settings

- To determine θs, We sampled one interaction for each user as the validation data and tuned θs on it.
- Negative sampling with logistic loss : we sampled four negative instances per positive instance.
- We initialized model parameters with a Gaussian distribution(μ = 0, σ = 0.01), optimizing the model with mini-batch Adam.
- Batch size : [128, 256, 512, 1024] Learning rate : [0.0001 ,0.0005, 0.001, 0.005]
- ***Predictive factors*** (# of nodes in last layers, a.k.a. model capacity) : [8, 16, 32, 64]
- Without special mention, we employed three hidden layers for MLP.

  i.e., If the size of predictive factors is 8, then the architecture of the neural CF layers is 32→16 →8, and the embedding size is 16.
- For the NeuMF with pre-training, α was set to 0.5 $\mathbf{h} \leftarrow \begin{bmatrix} \alpha \mathbf{h}^{GMF} \\ (1-\alpha)\mathbf{h}^{MLP} \end{bmatrix}$

Three Research Questions

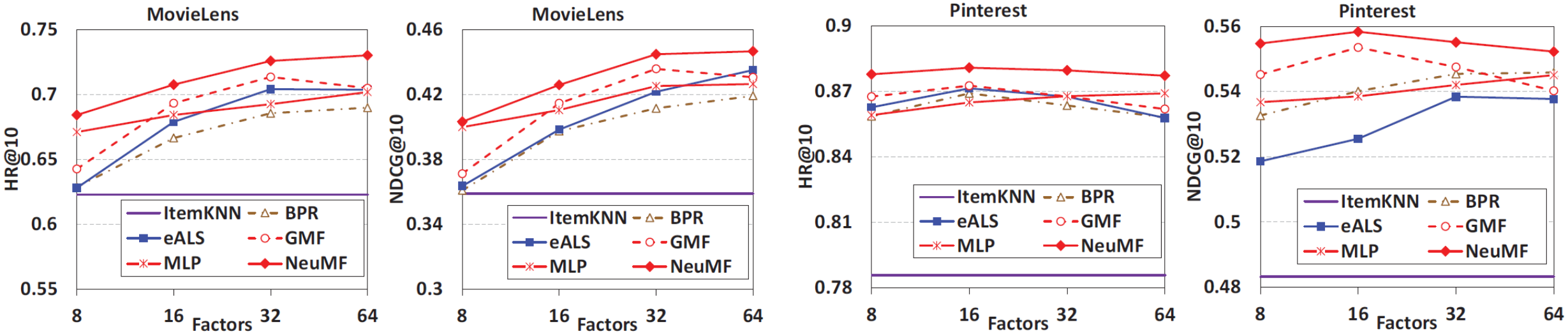
RQ1. Do our proposed NCF methods outperform the state-of-the-art implicit collaborative filtering methods?

RQ2. How does our proposed optimization framework (log loss with negative sampling) work for the recommendation task?

RQ3. Are deeper layers of hidden units helpful for learning from user–item interaction data?

RQ1. Do our proposed NCF methods outperform the state-of-the-art implicit collaborative filtering methods?

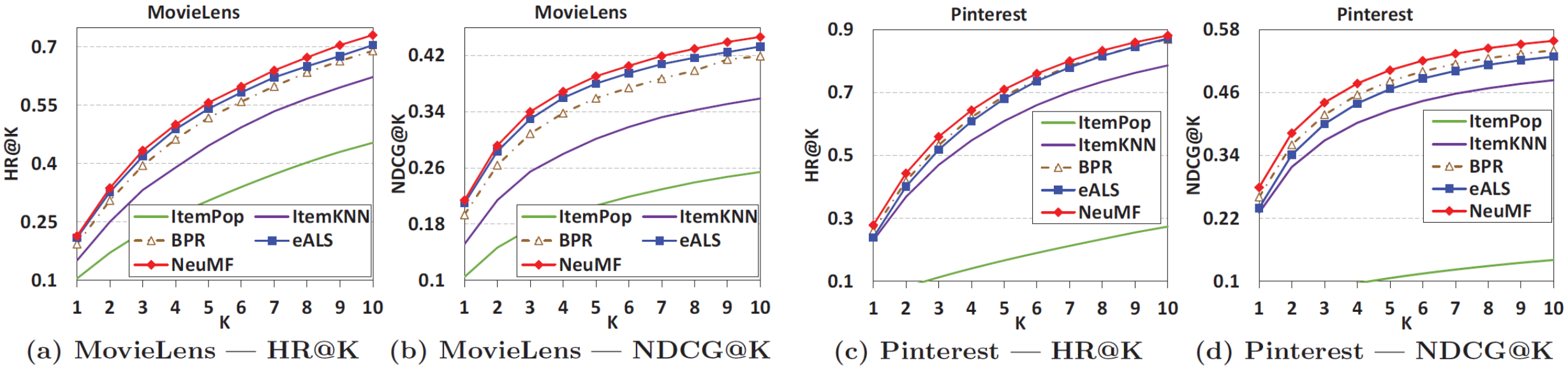
Figure 4: Performance of HR@10 and NDCG@10 *w.r.t.* the number of predictive factors on the two datasets.



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

- This indicates the high expressiveness of NeuMF by fusing the linear MF and non-linear MLP models.
- MLP can be further improved by adding more hidden layers (see Section 4.4)
- Although GMF suffers from overfitting for large factors, its best performance obtained is better than that of eALS.
- GMF shows consistent improvements over BPR, admitting the effectiveness of the classification-aware log loss for the recommendation task.

Figure 5: Evaluation of Top-K item recommendation where K ranges from 1 to 10 on the two datasets.



(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

- NeuMF has best performance among all.
- BPR can be a strong performer for ranking performance owing to its pairwise ranking-aware learner.
- The neighbor-based ItemKNN underperforms model-based methods.
- ItemPop performs the worst, indicating the necessity of modeling users' personalized preferences, rather than just recommending popular items to users.

Utility of Pre-training

Table 2: Performance of NeuMF with and without pre-training.

	With Pre-training		Without Pre-training	
Factors	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens				
8	0.684	0.403	**0.688**	**0.410**
16	**0.707**	**0.426**	0.696	0.420
32	**0.726**	**0.445**	0.701	0.425
64	**0.730**	**0.447**	0.705	0.426
Pinterest				
8	**0.878**	**0.555**	0.869	0.546
16	**0.880**	**0.558**	0.871	0.547
32	**0.879**	**0.555**	0.870	0.549
64	**0.877**	**0.552**	0.872	0.551

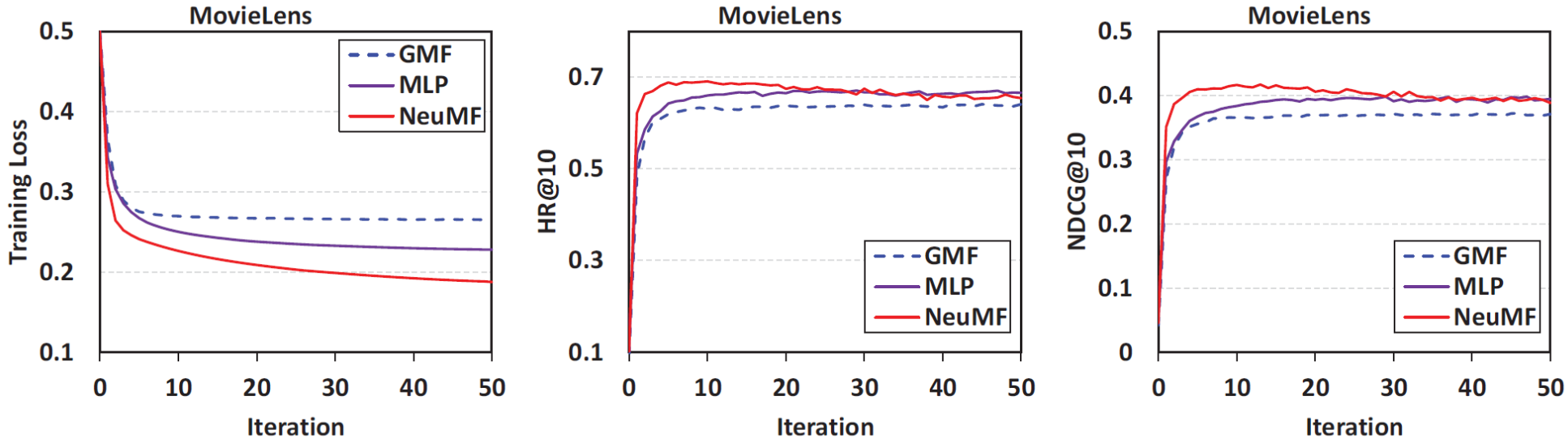
- For NeuMF without pre-training, we used the Adam to learn it with random initializations.
- This result justifies the usefulness of our pre-training method for initializing NeuMF.

4.3 Log Loss with Negative Sampling (RQ2)

RQ2. How does our proposed optimization framework (log loss with negative sampling) work for the recommendation task?

4.3 Log Loss with Negative Sampling (RQ2)

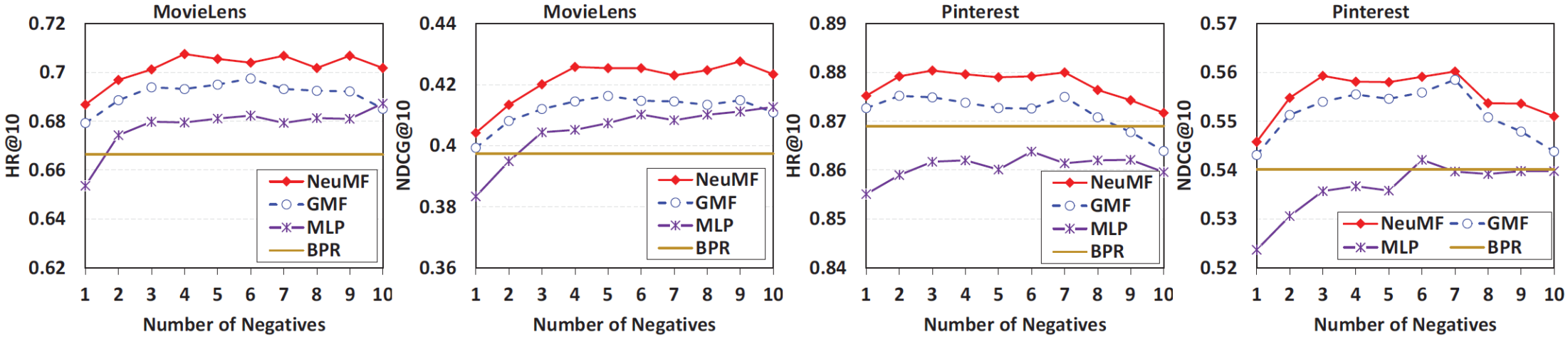
Figure 6: Training loss and recommendation performance of NCF methods *w.r.t.* the number of iterations on MovieLens (factors=8)



(a) Training Loss (b) HR@10 (c) NDCG@10

- The most effective updates are occurred in the first 10 iterations, and more iterations may overfit a model.
- Among the three NCF methods, NeuMF achieves the lowest training loss, followed by MLP, and then GMF.
- This is empirical evidence for the rationality and effectiveness of optimizing the log loss for learning from implicit data.

Figure 7: Performance of NCF methods *w.r.t.* the number of negative samples per positive instance (factors=16).



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

- An advantages of pointwise log loss over pairwise objective functions is **the flexible sampling ratio for negative instances**.
- Just one negative sample per positive instance is insufficient to achieve optimal performance, and sampling more negative instances is beneficial.
- For both datasets, the optimal sampling ratio is around 3 to 6.
- Setting the sampling ratio too aggressively may adversely hurt the performance.

4.4 Is Deep Learning Helpful? (RQ3)

RQ3. Are deeper layers of hidden units helpful for learning from user–item interaction data?

4.4 Is Deep Learning Helpful? (RQ3)

Table 3: HR@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.452	0.628	0.655	0.671	**0.678**
16	0.454	0.663	0.674	0.684	**0.690**
32	0.453	0.682	0.687	0.692	**0.699**
64	0.453	0.687	0.696	0.702	**0.707**
Pinterest					
8	0.275	0.848	0.855	0.859	**0.862**
16	0.274	0.855	0.861	0.865	**0.867**
32	0.273	0.861	0.863	**0.868**	0.867
64	0.274	0.864	0.867	0.869	**0.873**

Table 4: NDCG@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.253	0.359	0.383	0.399	**0.406**
16	0.252	0.391	0.402	0.410	**0.415**
32	0.252	0.406	0.410	**0.425**	0.423
64	0.251	0.409	0.417	0.426	**0.432**
Pinterest					
8	0.141	0.526	0.534	0.536	**0.539**
16	0.141	0.532	0.536	0.538	**0.544**
32	0.142	0.537	0.538	0.542	**0.546**
64	0.141	0.538	0.542	0.545	**0.550**

05. RELATED WORK

06 Related Work

- Explicit feedback에 집중한 초기 Recommendation system 연구

❖ R. Salakhutdinov, A. Mnih, and G. Hinton. *Restricted boltzmann machines for collaborative filtering*. In *ICDM*, pp.791–798, 2007.

❖ B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. *Item-based collaborative filtering recommendation algorithms*. In *WWW*, pp.285–295, 2001.

- Implicit feedback을 이용한 Item 추천에 대한 latent factor models를 조정하기 위해 Uniform weighting을 적용하는 초기 연구

❖ Y. Hu, Y. Koren, and C. Volinsky. *Collaborative filtering for implicit feedback datasets*. In *ICDM*, pages 263–272, 2008.

❖ S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. *BPR: Bayesian personalized ranking from implicit feedback*. In *UAI*, pages 452–461, 2009.

06 Related Work

저자는 앞에서 언급한 관련 논문들 중 대다수가 explicit ratings와 observed data에 대해서만 초점을 맞추었다고, 결과적으로 그 모델들은 positive-only implicit data로부터 users의 선호도를 학습하는 데에는 실패했다고 지적한다.

- Implicit feedback을 사용하는 CF에 대해 Autoencoder를 제안하는 논문

❖ Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. *Collaborative denoising auto-encoders for top-n recommender systems*. In *WSDM*, pages 153–162, 2016.

- 본 논문과 유사한 구조의 신경망 모델링을 사용한 논문들

❖ R. Socher, D. Chen, C. D. Manning, and A. Ng. *Reasoning with neural tensor networks for knowledge base completion*. In *NIPS*, pages 926–934, 2013.

❖ Heng-Tze Cheng et al., *Wide & Deep Learning for Recommender Systems*, DLRS 2016: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp.7-10, 2016.

06. CONCLUSION AND FUTURE WORK

06 Conclusion and Future Work

- In this work, we explored neural network architectures for collaborative filtering.
- We devised a general framework NCF and proposed three instantiations — GMF, MLP and NeuMF — that model user–item interactions in different ways.
- In future, we will study pairwise learners for NCF models and extend NCF to model auxiliary information, such as user reviews, knowledge bases, and temporal signals.
- While existing personalization models have primarily focused on individuals, it is interesting to develop models for groups of users, which help the decision-making for social groups.
- Moreover, we are particularly interested in building recommender systems for multi-media items, an interesting task but has received relatively less scrutiny in the recommendation community.
- Another emerging direction is to explore the potential of recurrent neural networks and hashing methods or providing efficient online recommendation.

07. REFERENCES

08 References

- **Xiangnan He et al., *Neural Collaborative Filtering*, WWW '17: Proceedings of the 26th International Conference on World Wide Web, pp.173-182, 2017.**
- Mingsheng Fu, Hong Qu, Zhang Yi, Li Lu, Yongsheng Liu, *A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System*, IEEE Transactions on Cybernetics, Vol 49, No. 3., pp.1084-1096, 2019.
- Paul Covington, Jay Adams, Emre Sargin, *Deep Neural Networks for YouTube Recommendations*, RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems, pp.191-198, 2016.
- Heng-Tze Cheng et al., *Wide & Deep Learning for Recommender Systems*, DLRS 2016: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp.7-10, 2016.
- Steffen Rendle et al., *BPR: Bayesian Personalized Ranking from Implicit Feedback*, UAI '09: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp.452-461, 2009.
- J. Leskovec, A. Rajaraman, J D. Ullman, *Mining of Massive Datasets*, 2nd; Cambridge University Press, 2016.
- Charu C. Aggarwal, *Recommender Systems: The Textbook*; Springer, 2016.

Q n A