NCF

Neural Collaborative Filtering

CONTENTS

01

Introduction

01. Introduction

Why NCF

연구 진행 이유 = MF is not sufficient for implicit data

1. user-item 간 latent feature modeling

2. Show MF $\subset$ NCF

3. NCF's nonlinearity

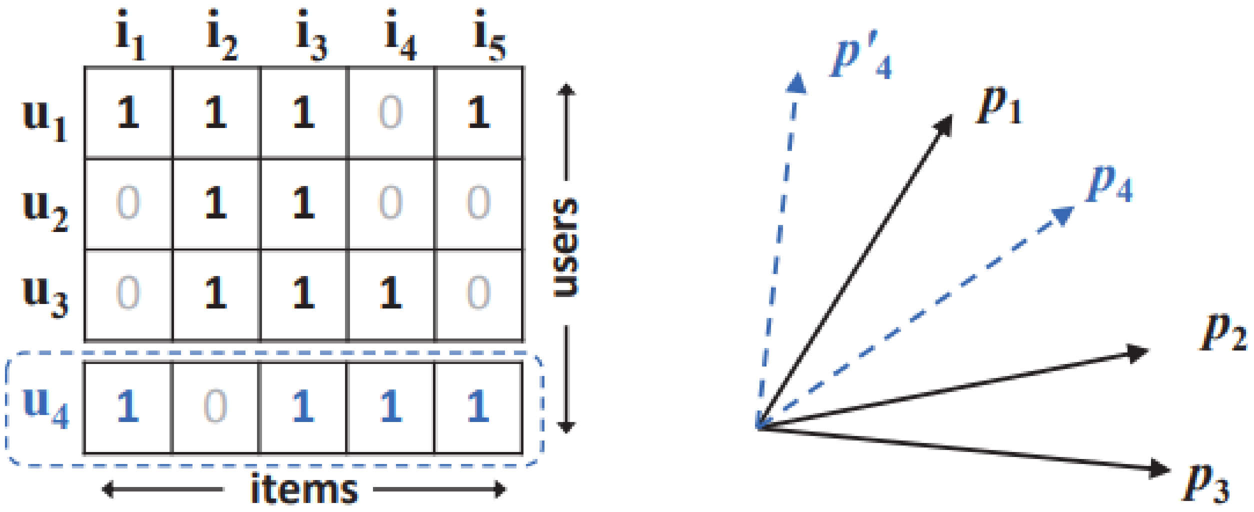
02

PRELIMINARIES

02. preliminaries

Learn from Implicit data

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



02. preliminaries

Learn from Implicit data

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

02. preliminaries

Learn from Implicit data

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

How to get

1. Pointwise loss *

2. Pairwise loss

+) Listwise loss

MF

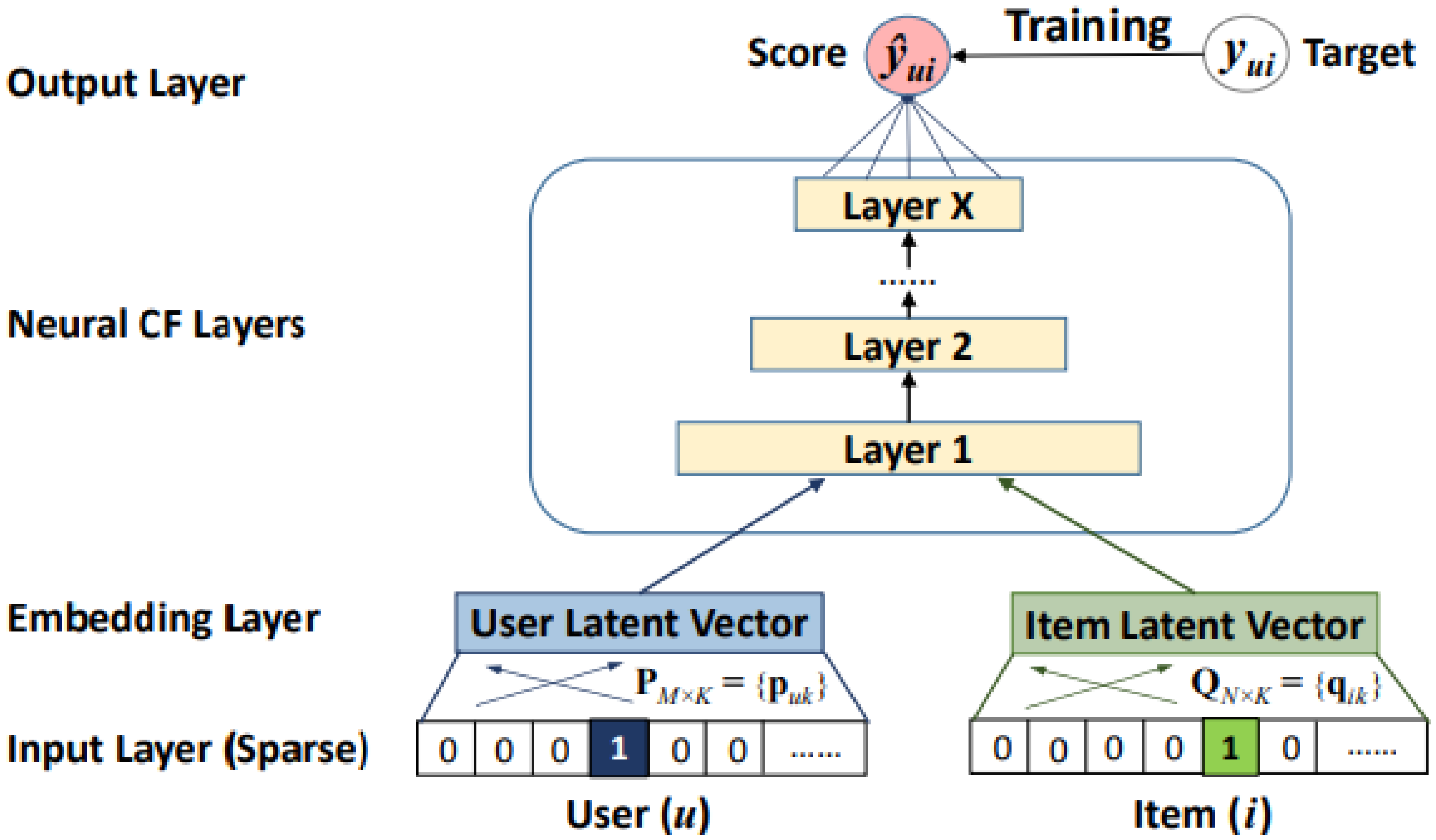
$$\hat{y}_{ui} = f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik},$$

03

NEURAL CF

03. Neural CF

General Framework



03. Neural CF

General Framework

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

03. Neural CF

General Framework

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

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

03. Neural CF

General Framework

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

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

GMF

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

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

03. Neural CF

MLP

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

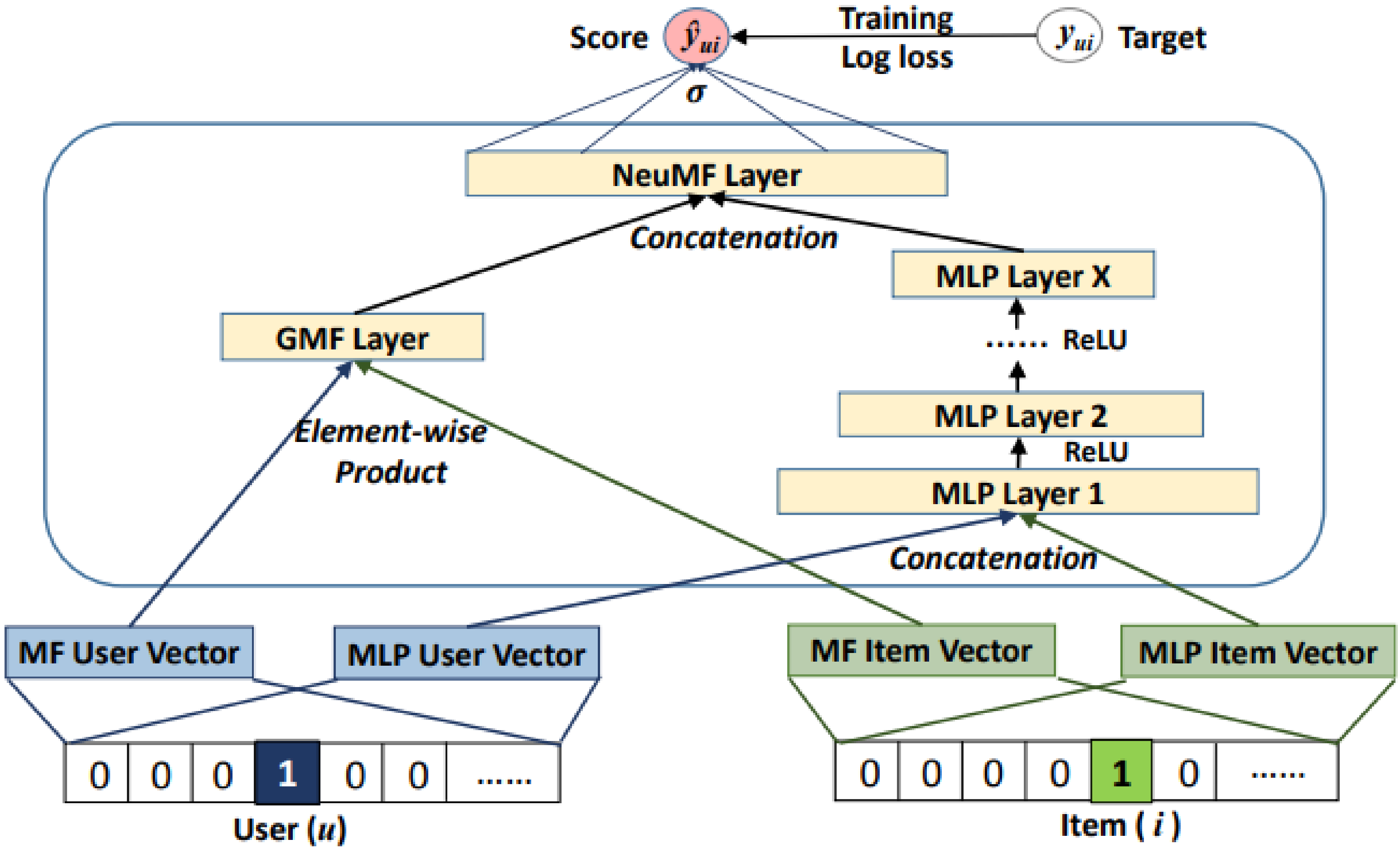
03. Neural CF

Fusion

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T a(\mathbf{p}_u \odot \mathbf{q}_i + \mathbf{W} \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix} + \mathbf{b}))$$

03. Neural CF

Fusion



03. Neural CF

Fusion

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G,$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),$$

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

04

Experiments

1. NCF is better than new implicit CF model

2. optimization framework fits well for recommendation

3. Deeper layer is good for find interaction

04. Experiments

Setting

Data set => MovieLens, Pinterest

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%

Score => leave-one-out (with sampling 100 random item)
HR + NDCG

$$Cumulative\ Gain(CG) = \sum_{i=1}^{n} relevance_i$$

$$g_{uj} = 2^{rel_{uj}} - 1$$

$$DCG = \frac{1}{m}\sum_{u=1}^{m}\sum_{j \in I_u, v_j \le L} \frac{g_{uj}}{\log_2(v_j + 1)}$$

Comparison model:

ItemPop

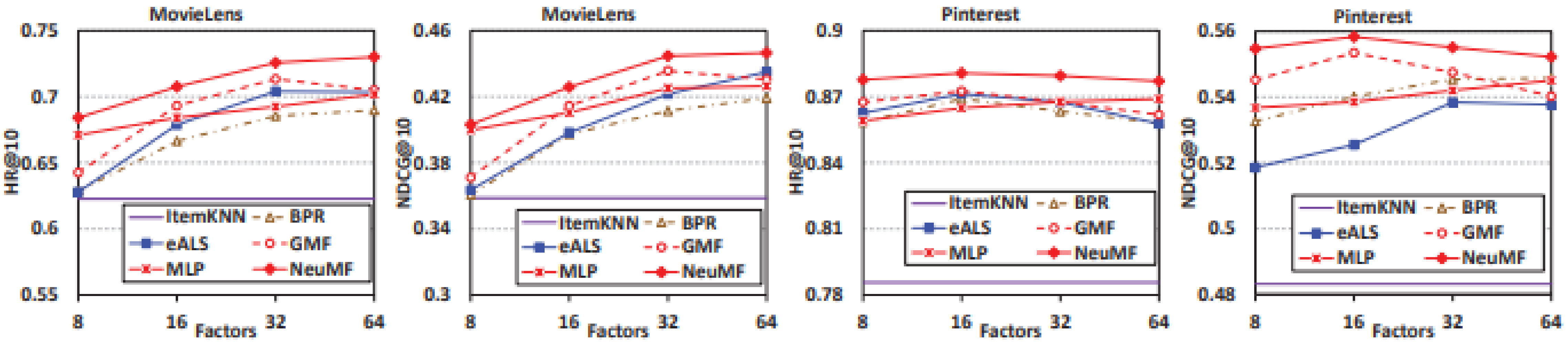
ItemKNN

BPR

eALS

04. Experiments

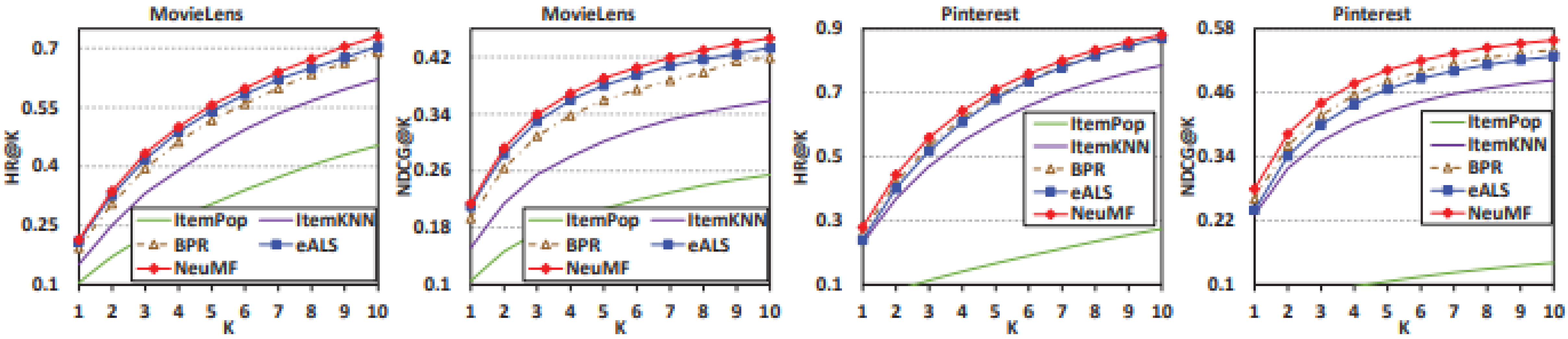
Result



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

04. Experiments

Result



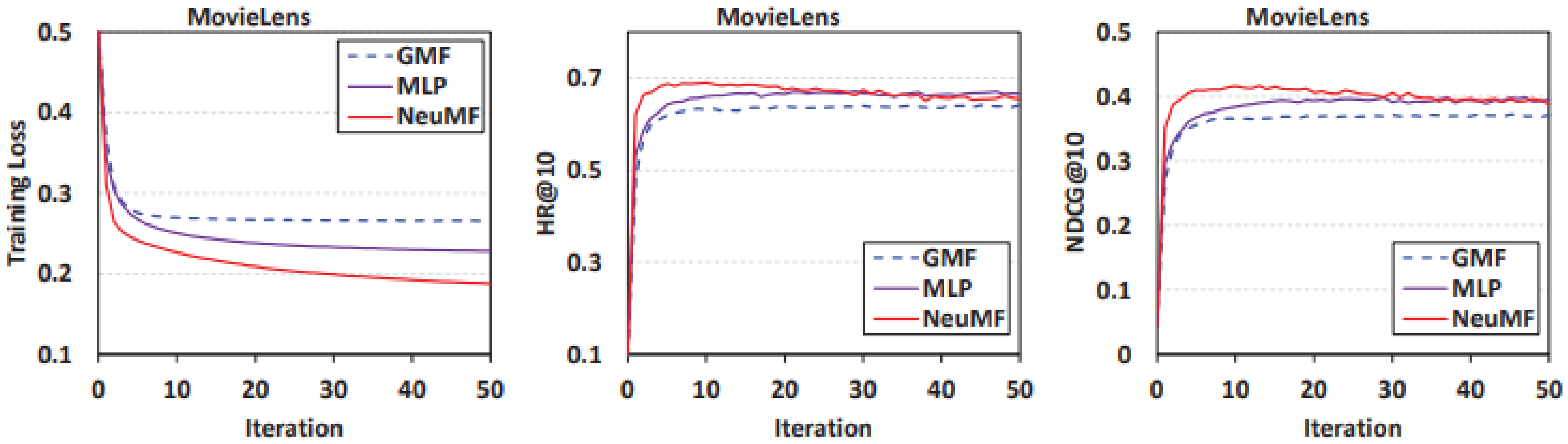
(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

Result

	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

04. Experiments

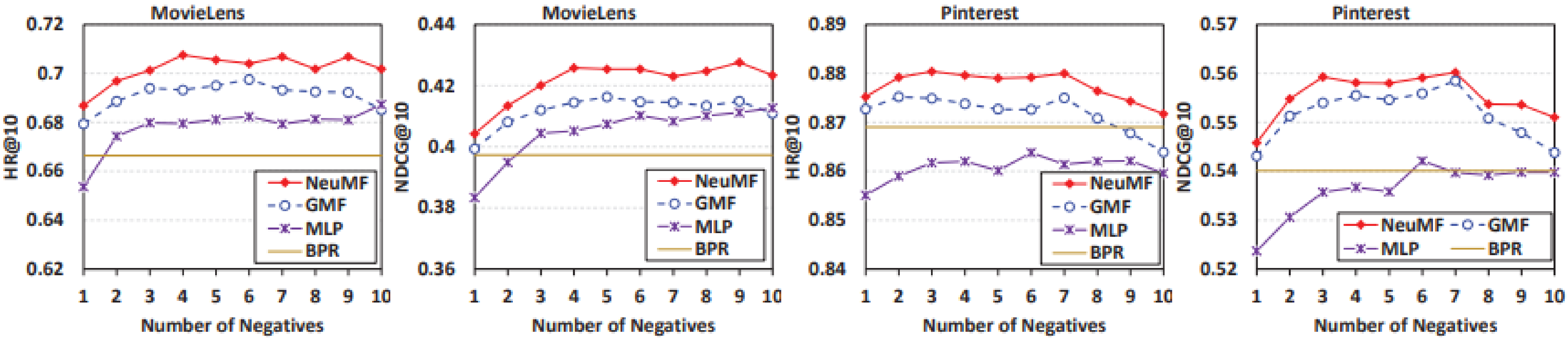
Result



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

04. Experiments

Result



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

04. Experiments

Result

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**

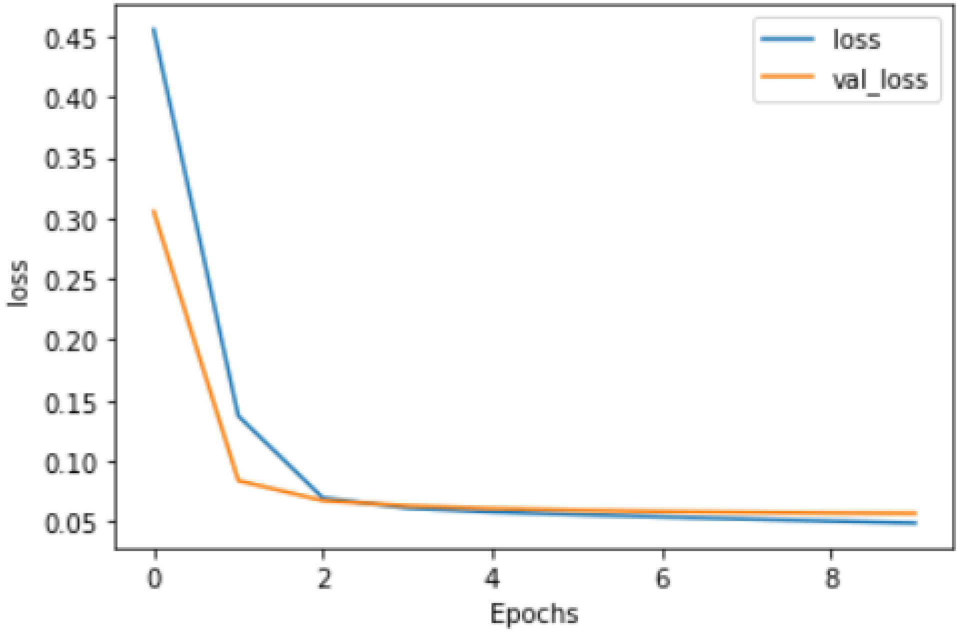
Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.253	0.359	0.383	0.399	**0.406**
16	0.252	0.391	0.402	0.410	**0.415**
32	0.252	0.406	0.410	**0.425**	0.423
64	0.251	0.409	0.417	0.426	**0.432**
Pinterest					
8	0.141	0.526	0.534	0.536	**0.539**
16	0.141	0.532	0.536	0.538	**0.544**
32	0.142	0.537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

04. Experiments

Result

```
User_embedding = tf.keras.layers.Embedding(n_users, embedding_dim,embeddings_regularizer=tf.keras.regularizers.l2(1e-6))(user_input)
```

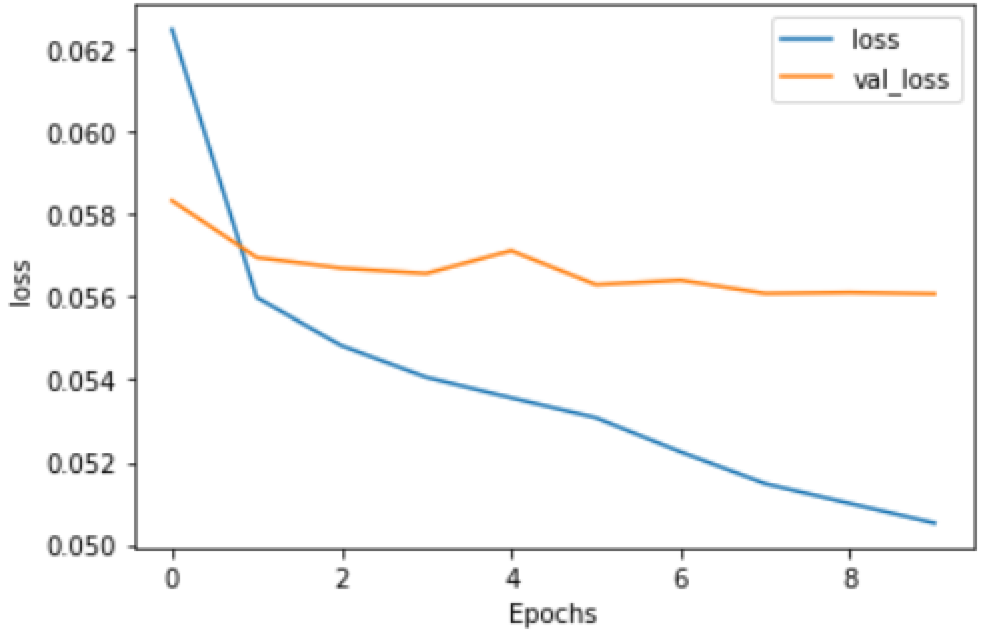
```
u = tf.keras.layers.Reshape((embedding_dim,))(User_embedding)
v = tf.keras.layers.Reshape((embedding_dim,))(Product_embedding)
s = tf.keras.layers.Dot(axes=1)([u, v])
```



04. Experiments

Result

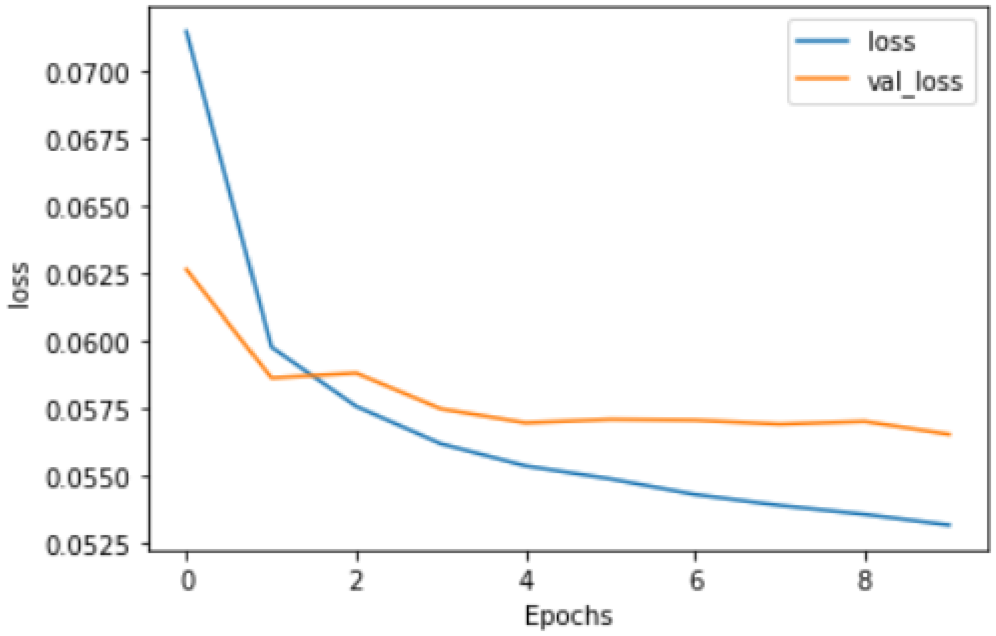
```
mlpl = tf.keras.layers.Dense(16, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(1e-6))(mlpl)
mlpl = tf.keras.layers.Dense(8, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(1e-6))(mlpl)

output_mlp = tf.keras.layers.Dense(1, activation='sigmoid', kernel_initializer="lecun_uniform")(mlpl)
```



04. Experiments

Result

```
ncf = tf.keras.layers.Concatenate(axis=1)([output_mlp, output_gmf])
output = tf.keras.layers.Dense(1, activation='sigmoid', kernel_initializer="he_normal")(ncf)
```



Thank you