Neural Collaborative Filtering

1.Introduction

2.Improvement

- Inner Product --> Neural network prediction

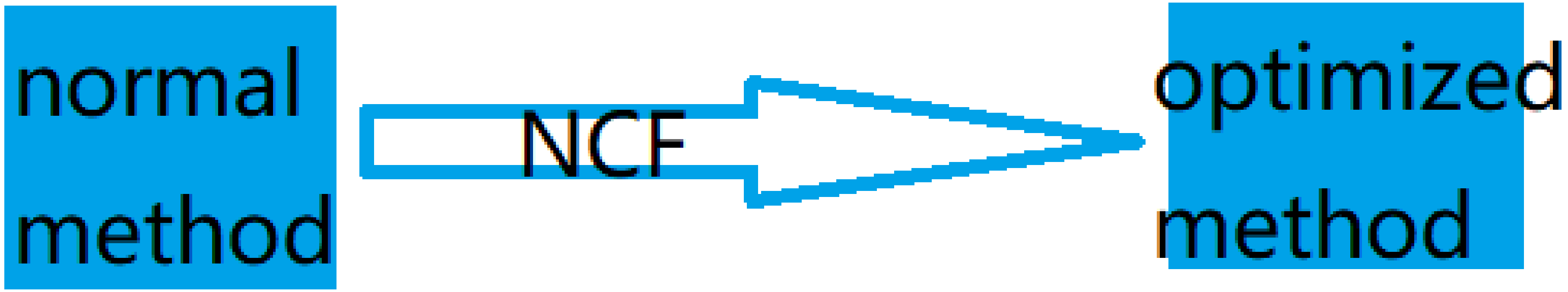
3.Core

- Generalized Matrix Factorization(GMF)
- Neural Network(MLP)
- Fusion

4.Result

Introduction

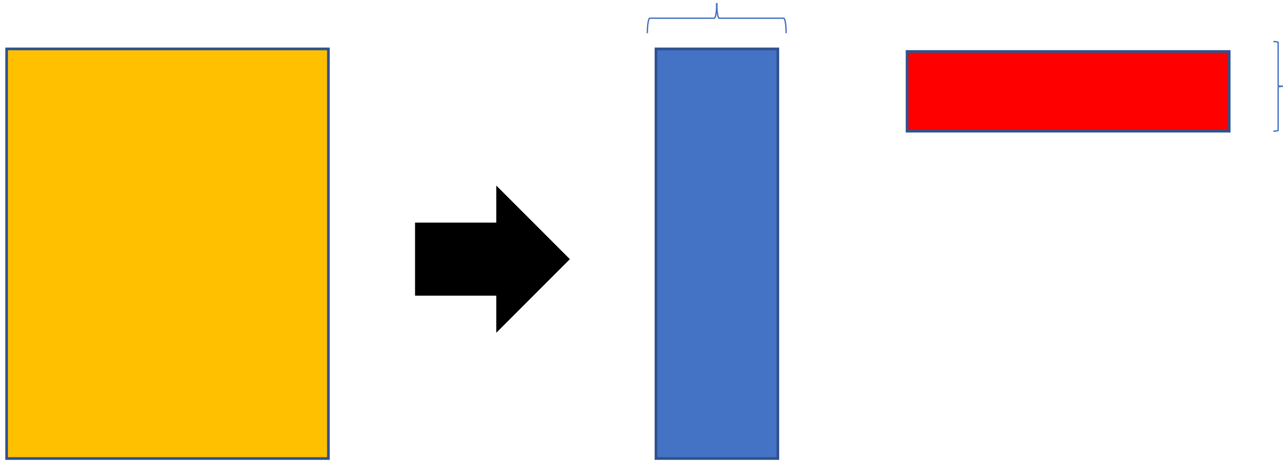
- NCF is an improvement for normal interaction method like MF.
- It uses the deep neural network(DNN) to learn the interaction function.
- Combines Linear(GMF) and non-Linear(MLP) model.



Improvement

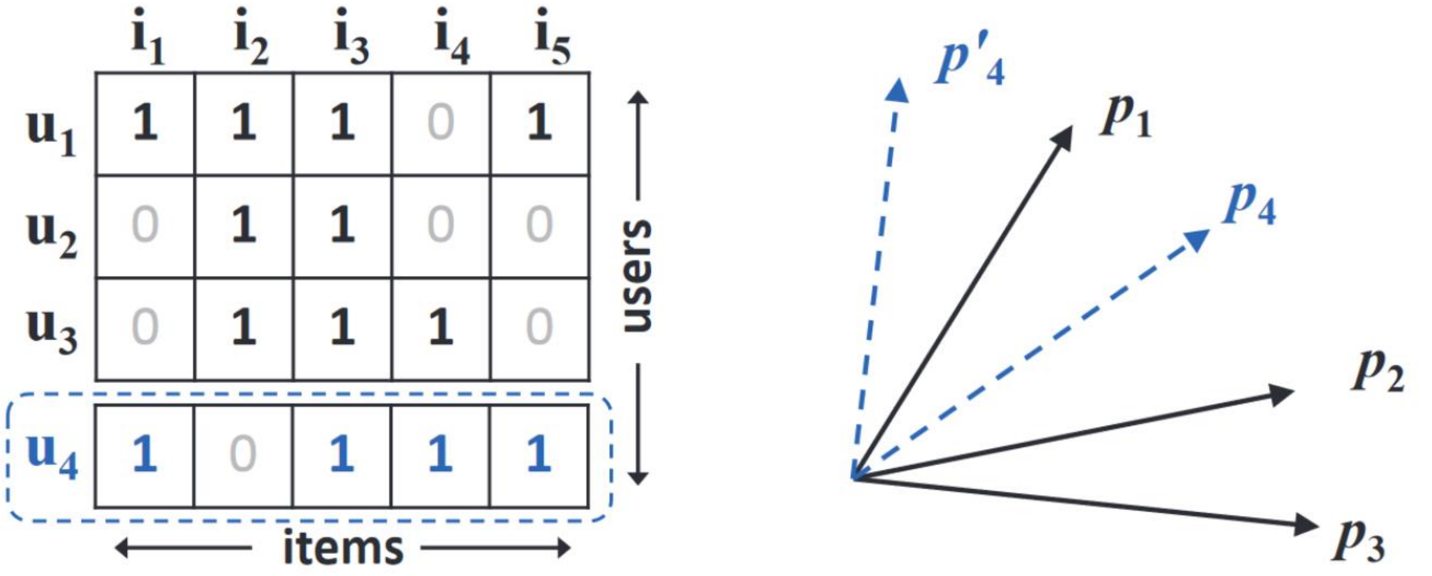
Matrix Factorization decompose a M*N user-item interaction matrix into a M*K user matrix and a K*N item matrix. And it is described that **we have projected the users and the items to the same K latent spaces.**

K latent spaces



If a user U and an item I are alike in the latent space, user U is prone to buy item I**. And we use the inner product to judge whether the user vector is close to the item vector**, that’s where using inner product as the possibility of interaction between a user and an item comes form

Improvement



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

But the latent space method has a limitation

$$s_{23}(0.66) > s_{12}(0.5) > s_{13}(0.4)$$

$$s_{41}(0.6) > s_{43}(0.4) > s_{42}(0.2),$$

We cannot lay U4 correctly

Using simple method like inner product or cosine index will lead to a great ranking loss

Improvement

Normal Matrix Factorization:

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

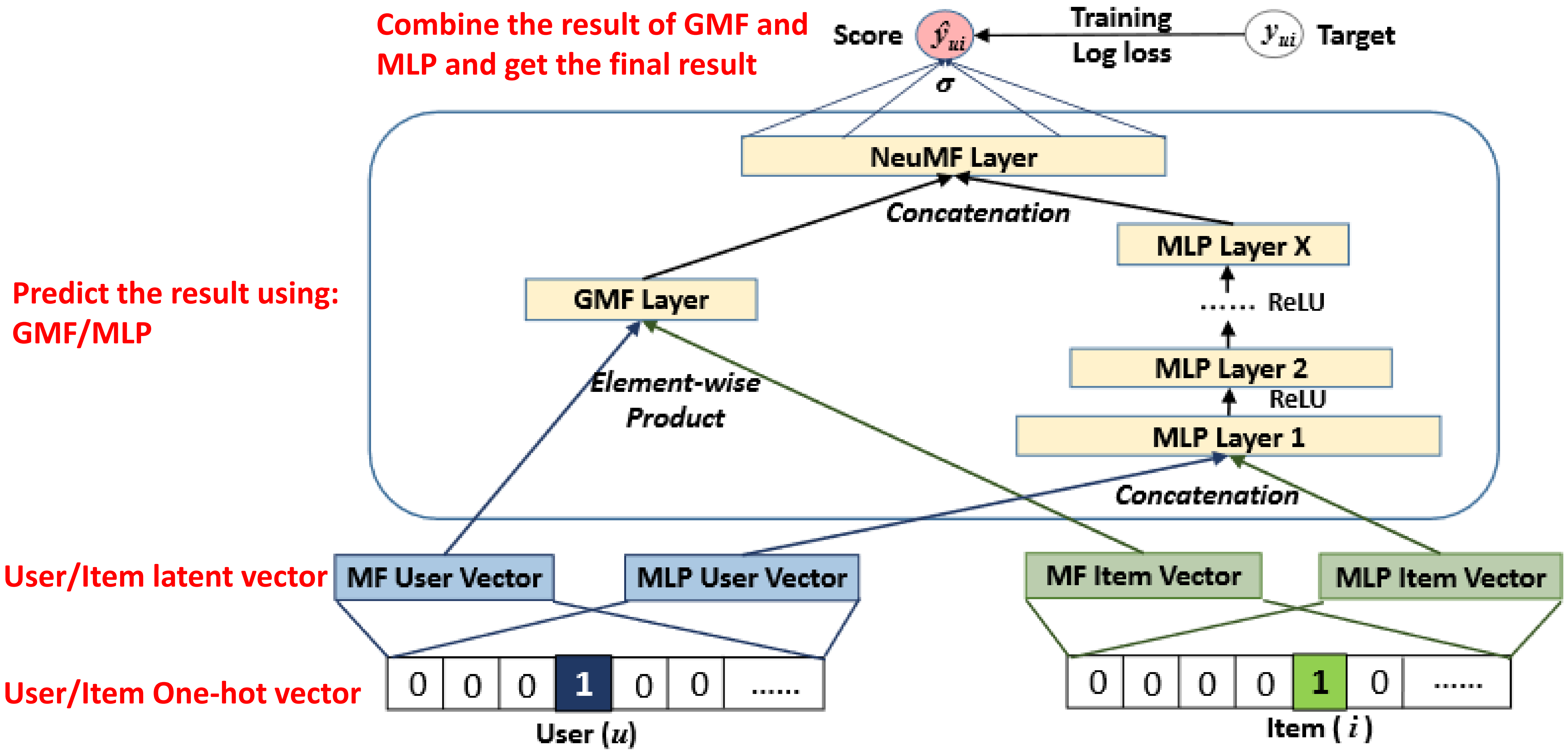
Use the inner production of user vector **U** and item vector **I** to predict the interaction between the user and the item.

NCF:

$$\hat{y}_{ui} = \text{NCF}(p_{uk} q_{ik})$$

Use Neural network to calculate the interaction between user **U** and item **I** on given input user vector U and item vector I

Core



Combine the result of GMF and MLP and get the final result
Predict the result using: GMF/MLP
User/Item latent vector
User/Item One-hot vector
Score
$\hat{y}_{ui}$
Training
Log loss
y_{ui}
Target
σ
NeuMF Layer
Concatenation
MLP Layer X
ReLU
GMF Layer
MLP Layer 2
ReLU
Element-wise Product
MLP Layer 1
Concatenation
MF User Vector
MLP User Vector
MF Item Vector
MLP Item Vector
0 0 0 1 0 0 ......
User (u)
0 0 0 0 1 0 ......
Item (i)

Core

$$\hat{y_{u,i}} = NCF(U_i^{GMF}, I_j^{GMF}, U_i^{MLP}, I_j^{MLP}, MLP - Net, h^{GMF}, h^{MLP}, \alpha)$$

$\hat{y}_{u,i}$ The predicted interaction between user u and item i (output)

$MLP - Net$ The MLP neural network

h^{GMF} A parameter for GMF part

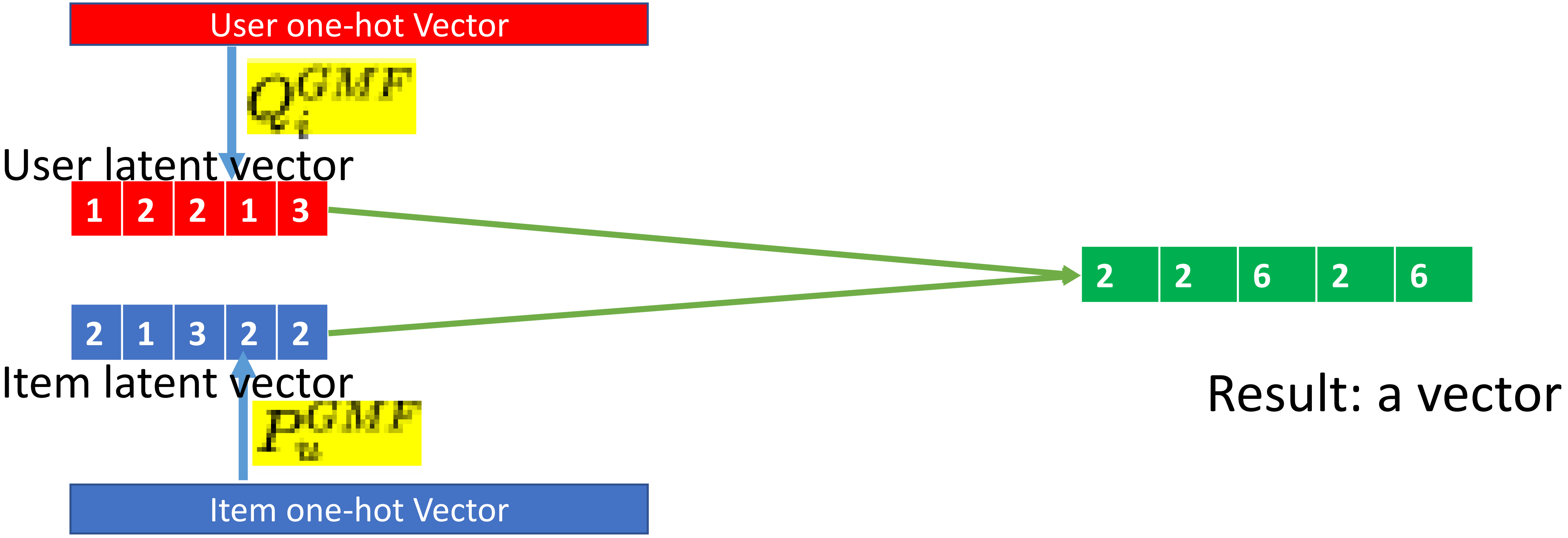
h^{MLP} A parameter for MLP part

α Voting weight for the fusion of two parts

Core

Generalized Matrix Factorization(Linear Part):

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

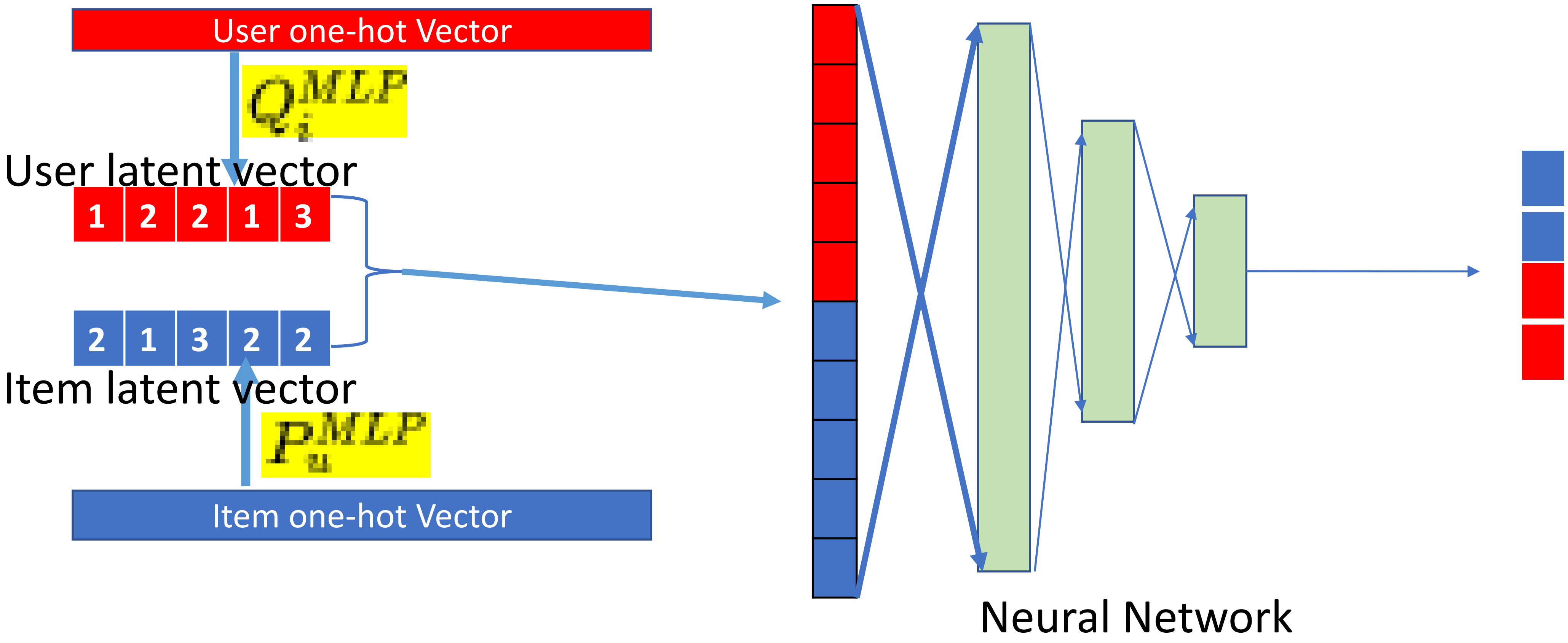


(inner product: the sum of each element in that vector, 18)

Core

Multi-Layer Perceptron(non-Linear Part):

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



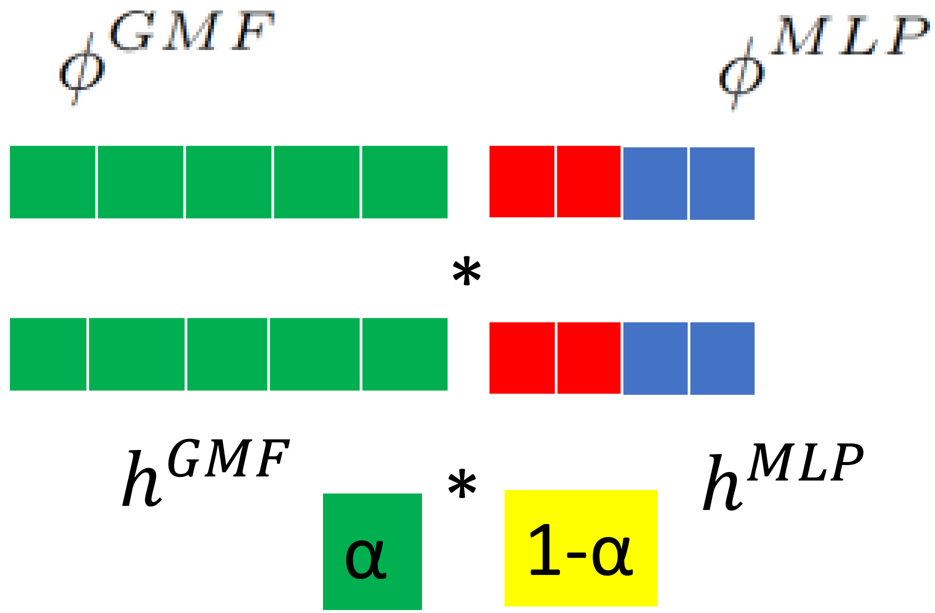
Core

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

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

Combines the result of both GMF and MLP using different weight(α)

$\hat{y}_{ui}$ = Sigmoid(sum( ))



Core

GMF,MLP: pre-training using Adam

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

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

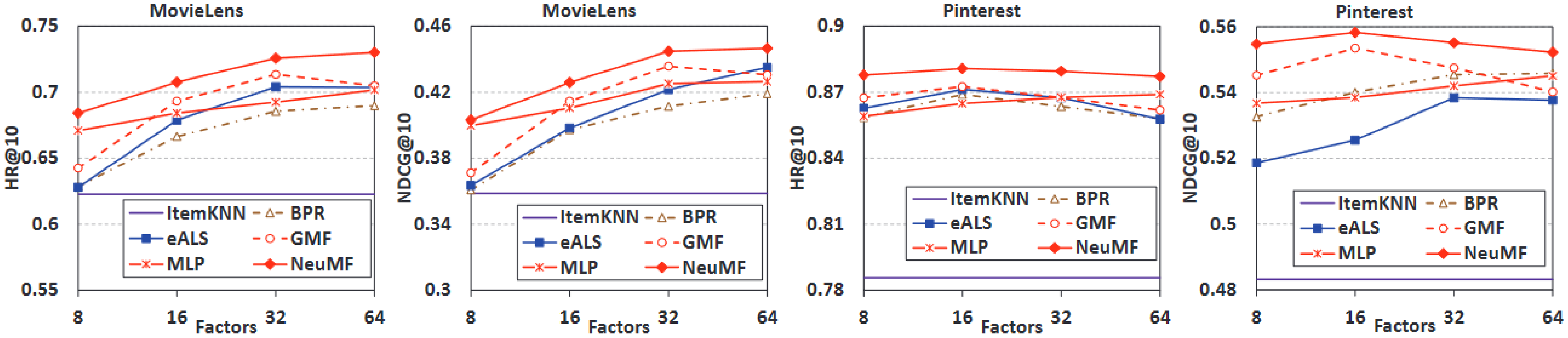
Use the trained GMF and MLP, then only tweak h,

using vanilla SGD, based on cross-entropy loss

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

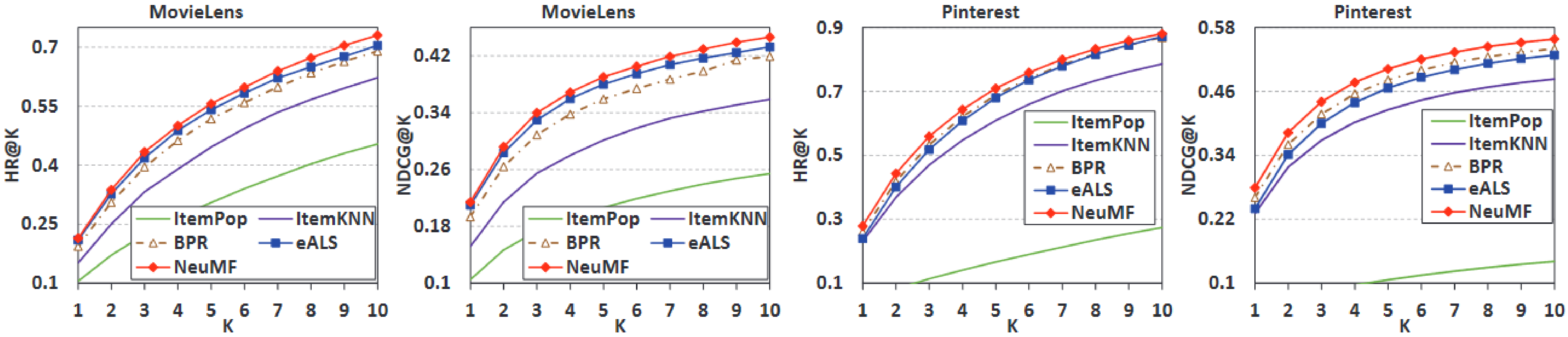
$$\begin{aligned} L &= -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j)\in\mathcal{Y}^-} \log(1-\hat{y}_{uj}) \\ &= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}). \end{aligned}$$

Result



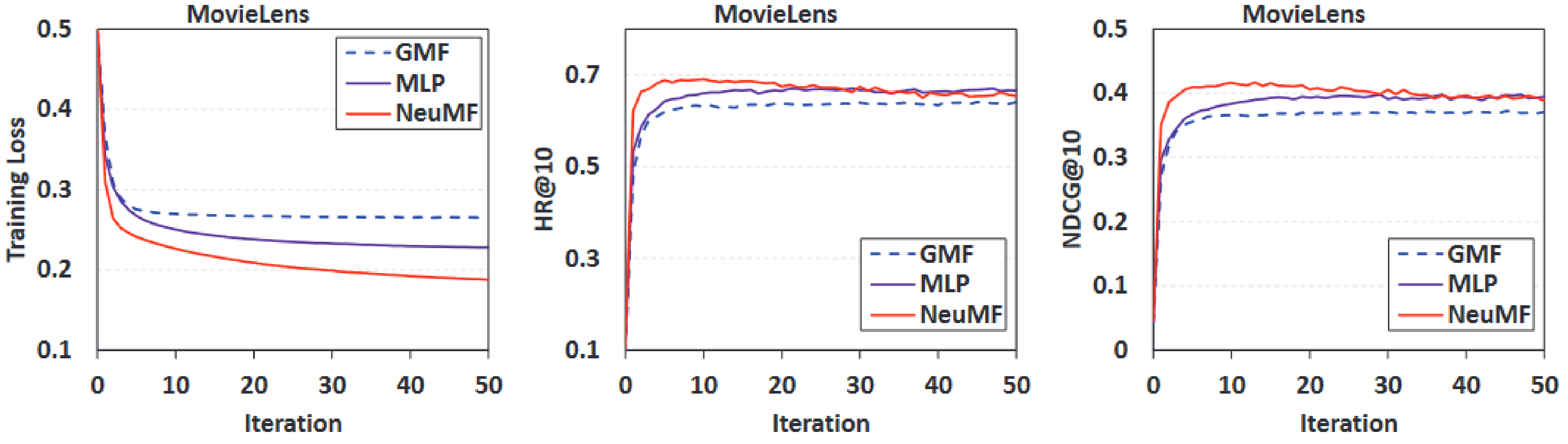
(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

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



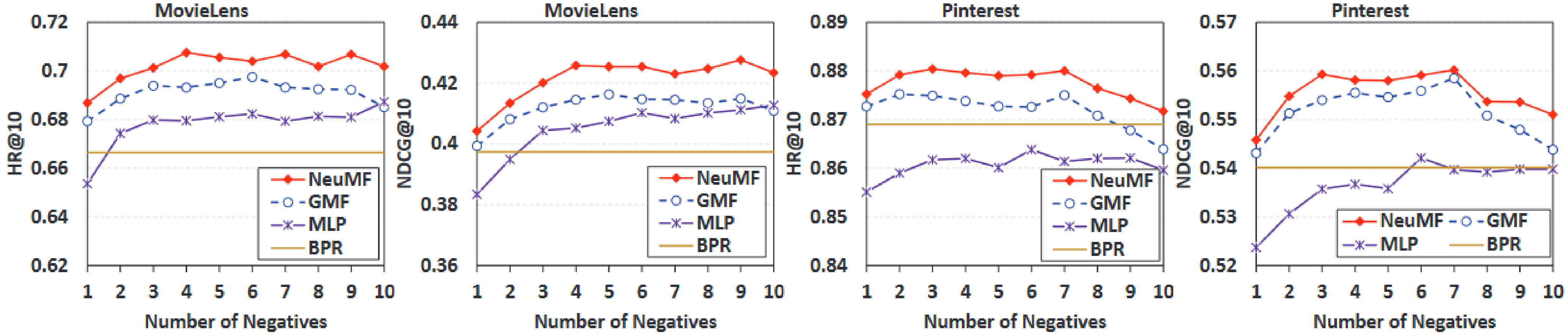
(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

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



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

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



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

Figure 7: Performance of NCF methods *w.r.t.* the number of negative samples per positive instance (factors=16). The performance of BPR is also shown, which samples only one negative instance to pair with a positive instance for learning.