Neural collaborative filtering

Xiangnan He et al.

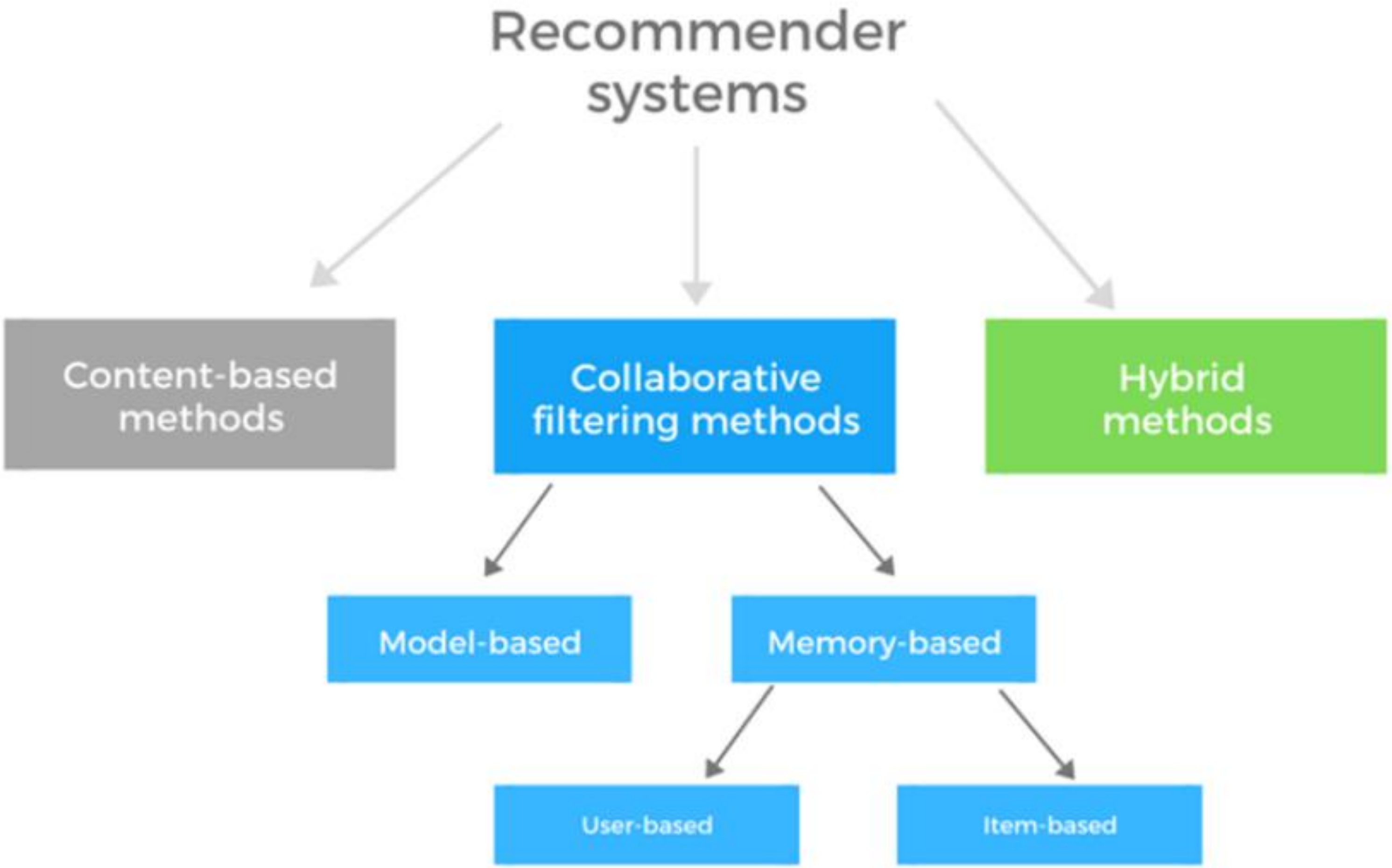
2017 International World Wide Web Conference Committee

연세대학교 응용통계학과 한솔지

contents

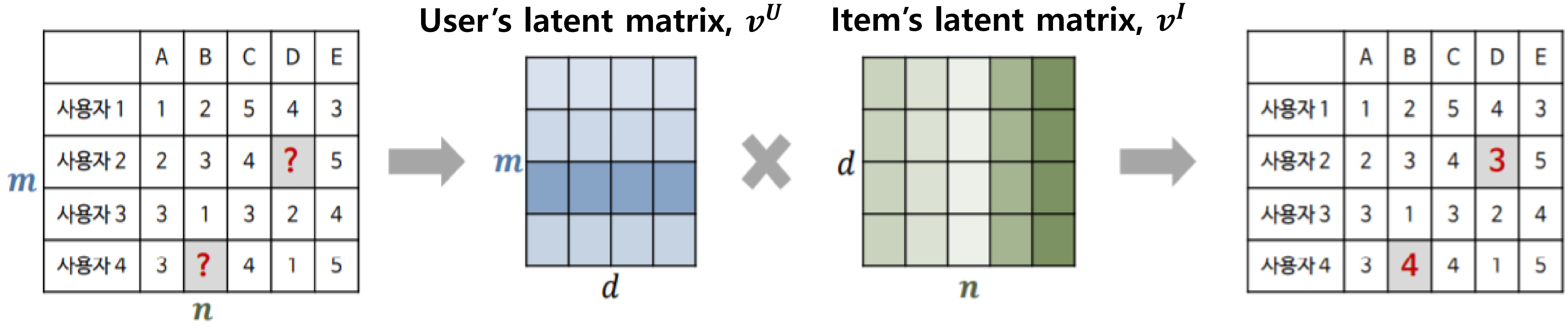
- Historical Overview
  - Introduction to recommender system
  - Matrix Factorization
- Proposed methods
  - Neural Collaborative Filtering(NCF)
  - Generalized Matrix Factorization (GMF)
  - Multi-Layer Perceptron (MLP)
- Experiment
- Conclusion

Introduction to Recommender System



Matrix Factorization (MF)

- MF is a **linear latent factor model**
  - Learn latent vector for each user, item : $\boldsymbol{v_u^U}, \boldsymbol{v_i^I}$
  - Affinity between user 'u' a and item 'i' : $\hat{y}_{ui} = < \boldsymbol{v_u^U}, \boldsymbol{v_i^I} >$



User 'u' interacted with item 'i'

Limitation of Matrix Factorization

- Inner product function can **limit the expressiveness of MF model**

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

새로운 U_4에 대해서
U_1, U_3, U_2순으로
유사하지만,
P_4 를 어디에 놓아도
유사도 관계를
만족 못함!

- Example :

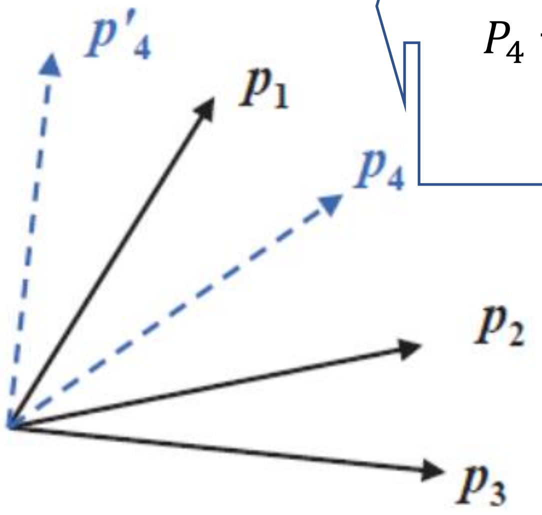
sim(u1, u2) = 0.5

sim(u3, u1) = 0.4
sim(u3, u2) = 0.66

	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

items ⟵⟶ users

(a) User-item matrix



(b) User latent space

Limitation of Matrix Factorization

- Inner product function can limit the expressiveness of MF model

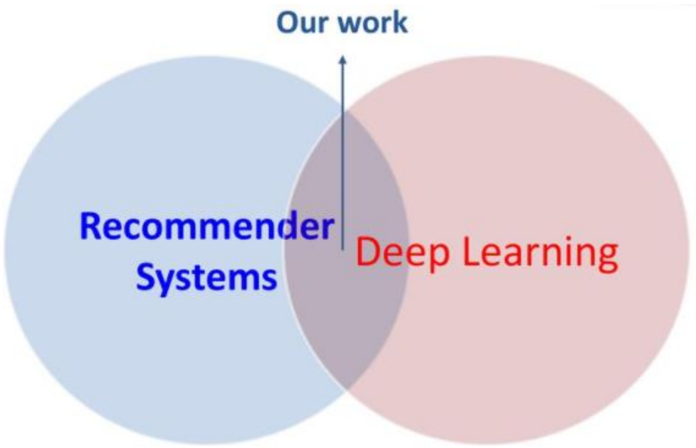
$$\hat{y}_{u,i} = f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u\mathbf{q}_i^T = \sum_{k=1}^{K} p_{uk}q_{ki}$$

→ can incur a large ranking loss for MF

- **Solution**

1) Using a large number of latent factors (overfitting)
2) **Learning the interaction function from data** (rather than the simple, fixed inner product)

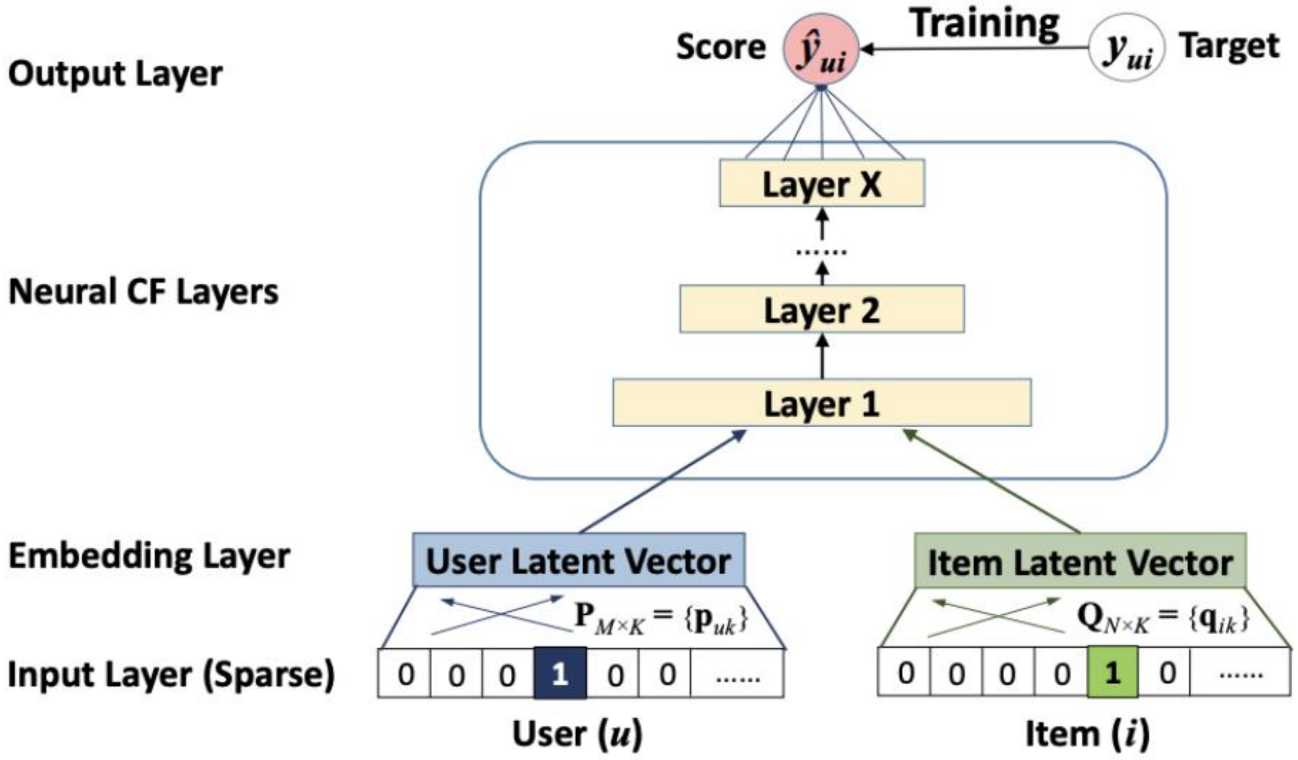
Proposed methods



- A Neural Collaborative Filtering(NCF)
    - ➢ learns the interaction function with a deep neural network.
- A NCF instance that generalizes the MF model (GMF).
- A NCF instance that models non-linearities with a multi-layer perceptron (MLP)
- A NCF instance NeuMF that fuses GMF and MLP.

Neural Collaborative Filtering(NCF)

- General Framework : $\hat{y}_{u,i} = f(p_u, q_i)$
  - NCF uses a multi-layer model to learn user-item interaction function



Generalized Matrix Factorization (GMF)

- NCF can express and generalize MF

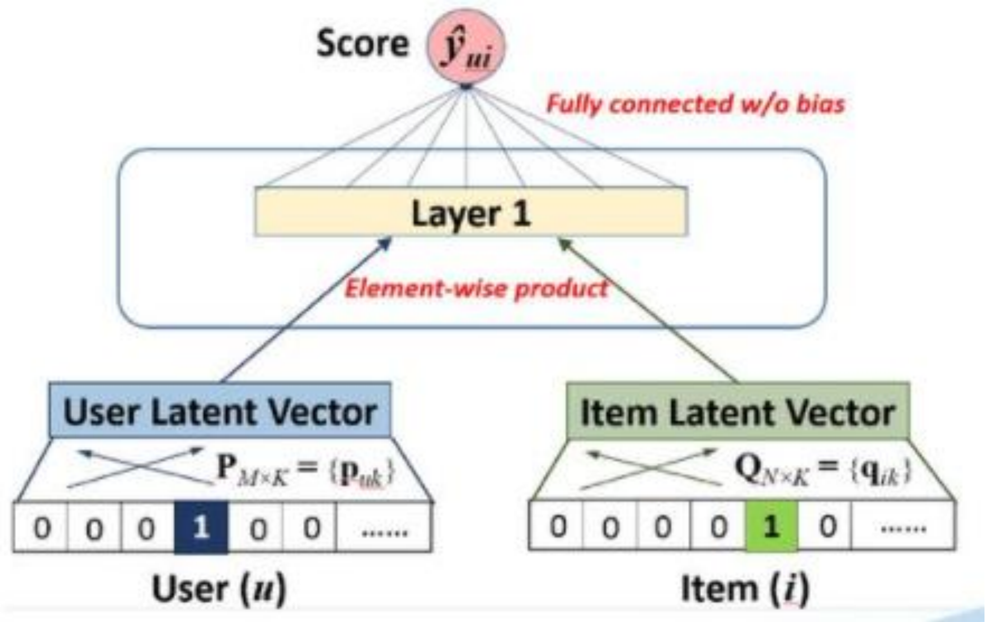
$$\hat{y}_{u,i} = f(p_u, q_i) = a_{out}\left(h^T \phi_1(p_u, q_i)\right) \quad \textit{where } \boldsymbol{\phi_1(p_u, q_i) = p_u \odot q_i}$$

1) MF : If $a_{out} = identity\ function$

$and\ \ h^T = [1, \dots, 1]_{1\times k}$

2) GMF: if a_{out} : sigmoid function

$and\ \ h^T = [h_1, \dots, h_k]$

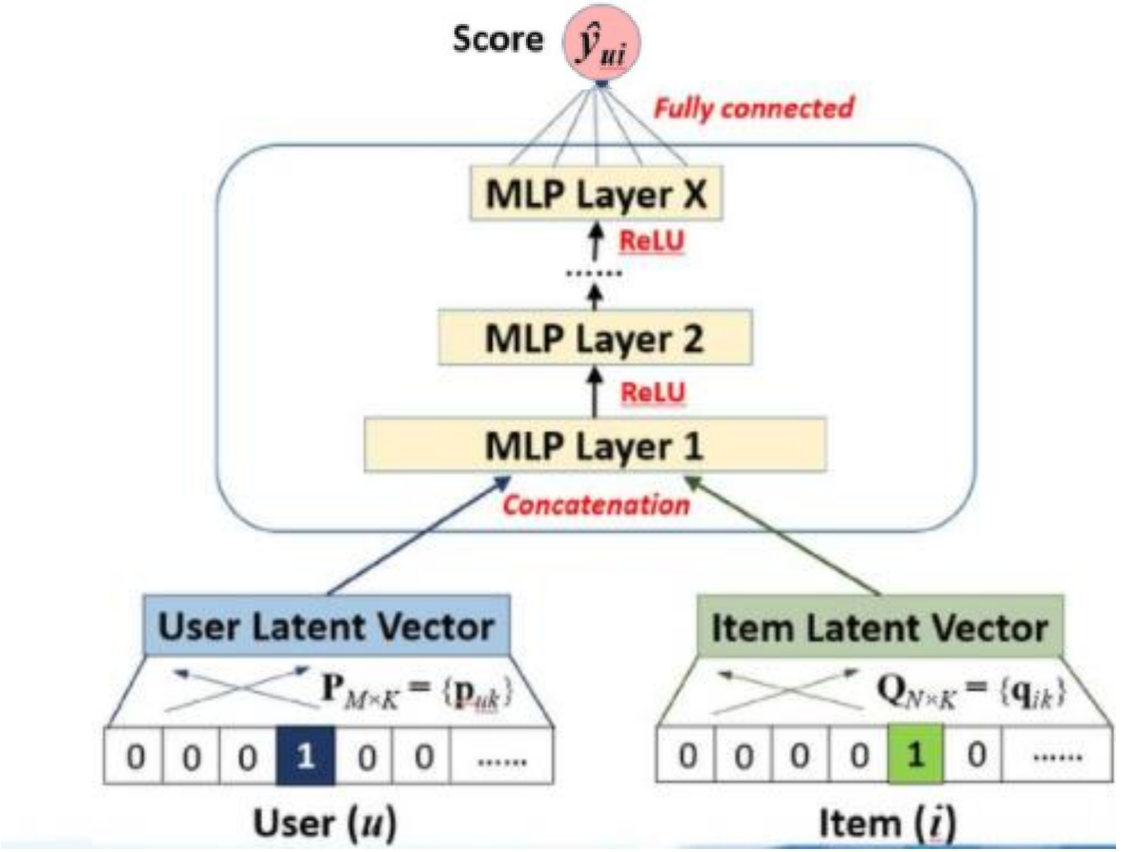


Multi-Layer Perceptron (MLP)

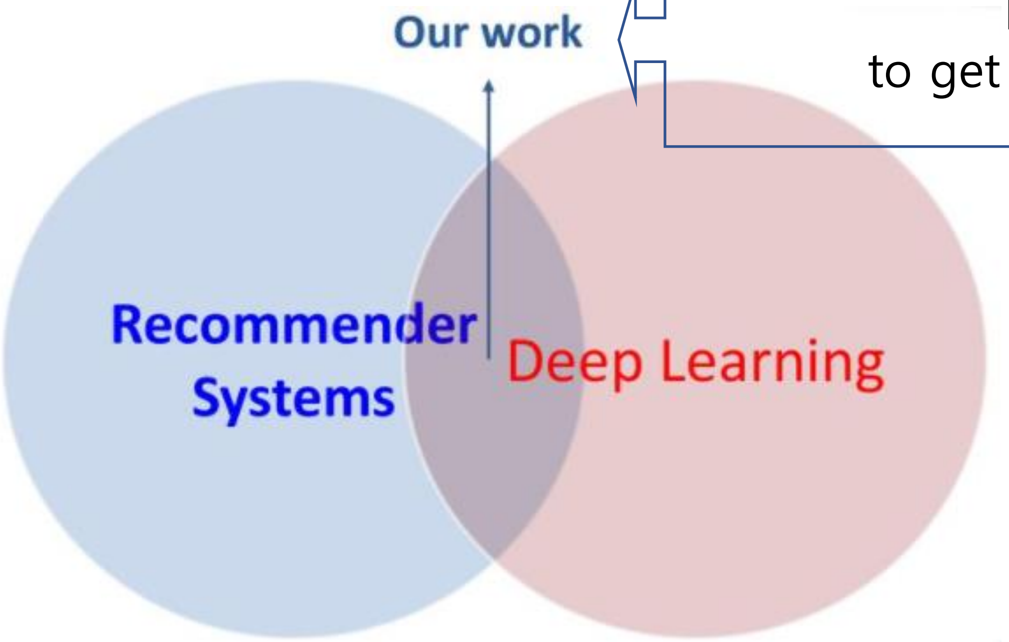
- More non-linearities to learn the interaction function! (concat + MLP)

  - Layer 1 $$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix},$$
  - Other Layer

$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2),$$
$$\cdots\cdots$$
$$\phi_L(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L),$$
$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),$$



MF vs MLP



Fuse two model
to get more powerful model!

$$\hat{y}_{u,i} = f(p_u, q_i)$$

MF : inner product
→ Latent factors are independent with each other
→ Good generalization ability for CF modelling

MLP : non-linear function
→ better representation ability
→ However, generalization ability is unknown!

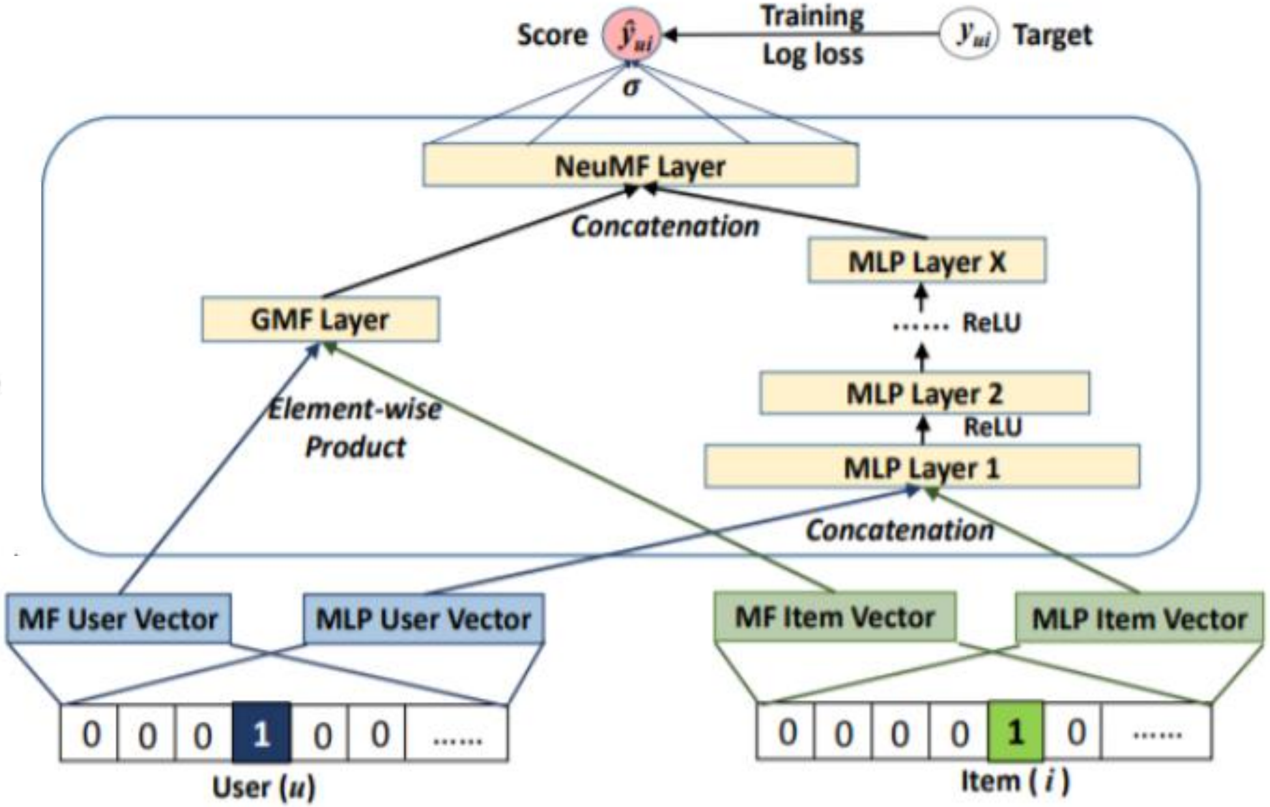
Fusion of GMF and MLP

- Neural Matrix Factorization (NeuMF)
    - GMF and MLP learn different sets of embeddings.

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

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

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



Experiment

- **Research questions**
  1) NCF outperform SOTA ?
  2) How does optimization framework work?
  3) Are deeper layers of hidden units helpful?

- **Data sets**
  1) Movielens
  2) 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%

- **Evaluation protocols**
  1) Randomly samples 100 items
  2) Ranking Evaluation Metrics
     - Hit Ratio (HR) , Normalized Discounted Cumulative Gain (NDCG)

Experiment

Ranking Evaluation Metrics

1) Hit Ratio (HR)

$$HR = \frac{\left|U_{hit}^{L}\right|}{\left|U_{all}\right|}$$

$$where \left|U_{hit}^{L}\right|: \#\ of\ users\ for\ which\ the\ correct\ answer\ is\ included\ in\ top\ L\ recomendation\ list,$$
$$\left|U_{all}\right| :\ the\ total\ number\ of\ users\ in\ the\ test\ data\ set.$$

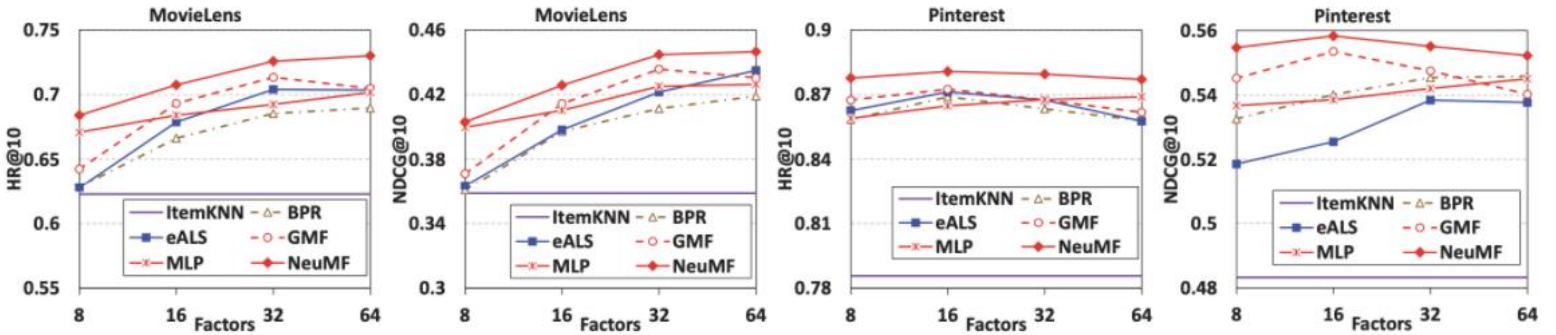
2) Normalized Discounted Cumulative Gain (NDCG)

$$NDCG(k) = \frac{DCG(k)}{IDCG(k)}$$

- $DCG = \Sigma_{i=1}^{\mathrm{k}} \frac{\mathrm{G_i}}{\log_2(i+1)}$, $IDCG = \Sigma_{i=1}^{|I(k)|} \frac{\mathrm{G_i}}{\log_2(i+1)}$
- Where $\mathrm{G_i}$: Gain for an item i (numerical ratings or binary in case of implicit data)
- k : a position *k* in the recommendation list , I(k) : the ideal list of items up to position k

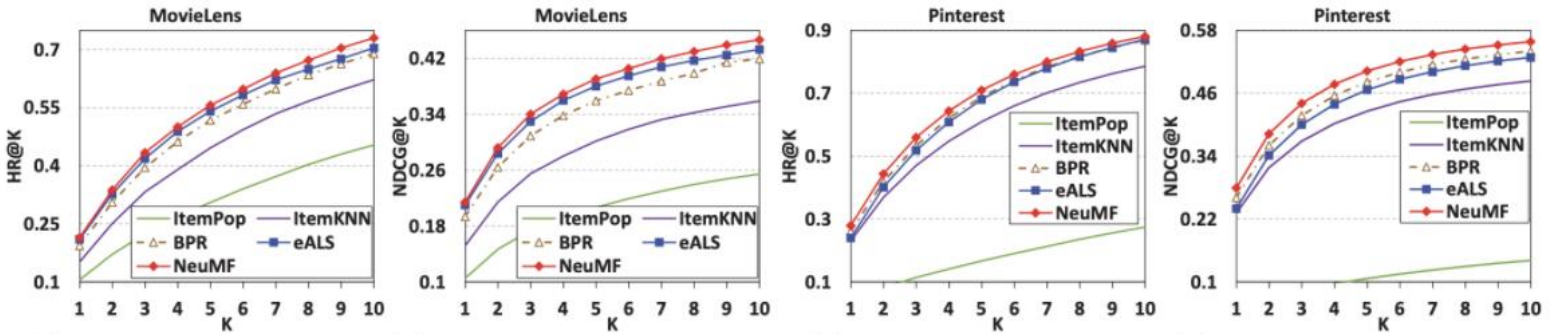
RQ(1) Performance comparison

- Baseline 방법들에 비해 NeuMF가 성능 좋음
- NCF 방법 중에서는 NeuMF > GMF ~= MLP 순으로 성능 보여짐



(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.

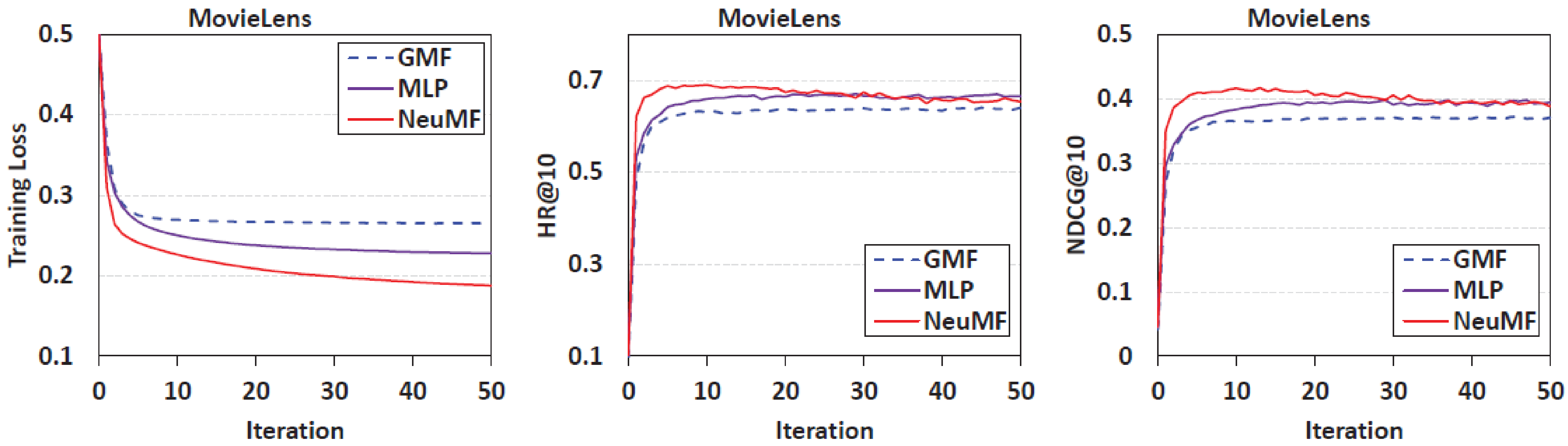


(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

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

RQ(2) Convergence Behavior

- Trade-off between representation ability and generalization ability
- NeuMF : 처음 10번 정도의 iteration 까지가 가장 효율적이고, 이후의 iteration은 데이터에 overfit



(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).

RQ(3) Is Deep Leaning Helpful?

- non-linearity 필요성
  - (동일한 모델 capability에서) MLP에서 non-linear layer의 개수를 늘리면, 성능이 더 좋아짐
  - 참고로 linear layer를 쌓는 것은 ReLu로 non-linear layer 쌓은 것보다 성능 안 좋음!

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**

Conclusion

- 기존 전통적인 collaborative filtering의 기본 구조 (user & item interaction 모델링)에 neural network 를 활용한 것
- Matrix factorization 의 한계를 지적하며, NN을 사용하여 user-item interaction의 구조를 linearity 에서 Non-linearity로 확장시킴
  - 기존 matrix factorization 의 generalized version
  - linear 커널 + non-linear 커널 → flexibility 증대
- 또한 collaborative filtering에서 Non-linearity의 필요성을 시사함