Neural Collaborative Filtering

Lab Intern 이준모

bubblego0217@kaist.ac.kr

Contents

Introduction

- Collaborative Filtering
  - Matrix Factorization (MF)
    - Interaction : Inner product of user and item latent vectors
    - Combine with neighbor-based models, Factorization Machines
    - Performance can be hindered by the interaction function
      - not be sufficient to capture the complex structure of interaction
- Using Deep Neural Network
  - Little work on Recommendation (Previous)
    - DNNs to model auxiliary information (ex. Textual description of items)
    - Still resorted to MF
  - Use DNNs for learning the interaction function from data

Implicit Feedback & Contribution

- Implicit Feedback
  - Can be tracked automatically & Much easier to collect
  - User Satisfaction is not observed & A natural Scarcity of Negative Feedback

- Contribution
  1. Neural Net Architecture to model latent features & General framework NCF
  2. MF = Specialization of NCF & Non-linearity with Multi-Layer Perceptron
  3. Effectiveness of NCF Approaches & Promise of Deep Learning for CF

Learning from Implicit Data

M and N : the number of users and items

$$\mathbf{Y} \in \mathbb{R}^{M \times N} \Longleftarrow y_{ui} = \begin{cases} 1, & \text{if interaction (user } u\text{, item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{array}{c|ccccc} & \mathbf{i}_1 & \mathbf{i}_2 & \mathbf{i}_3 & \mathbf{i}_4 & \mathbf{i}_5 \\ \hline \mathbf{u}_1 & 1 & 1 & 1 & 0 & 1 \\ \mathbf{u}_2 & 0 & 1 & 1 & 0 & 0 \\ \mathbf{u}_3 & 0 & 1 & 1 & 1 & 0 \\ \mathbf{u}_4 & 1 & 0 & 1 & 1 & 1 \end{array}$$

(a) user–item matrix

※ Interaction doesn't mean Preference

- Observed : at least interest
- Unobserved : Missing Data & Scarcity of Negative Feedback

Learning from Implicit Data

- Recommendation problem with Implicit Feedback
  - Estimating the scores of unobserved entries in Y
    - Abstracted as learning $\hat{y}_{ui} = f(u, i|\Theta)$
- Estimate parameters Θ
  - Follow the Machine Learning Paradigm – Optimizes an objective function
    - Pointwise Loss
      - $min \frac{1}{2}(\hat{y}_{ui} - y_{ui})^2$ (ex. OCCF)
    - Pairwise Loss
      - $\max(0, f(y_{unobs}) - f(y_{obs}) + \alpha)$ (ex. BPR)

Matrix Factorization

- Abstraction

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

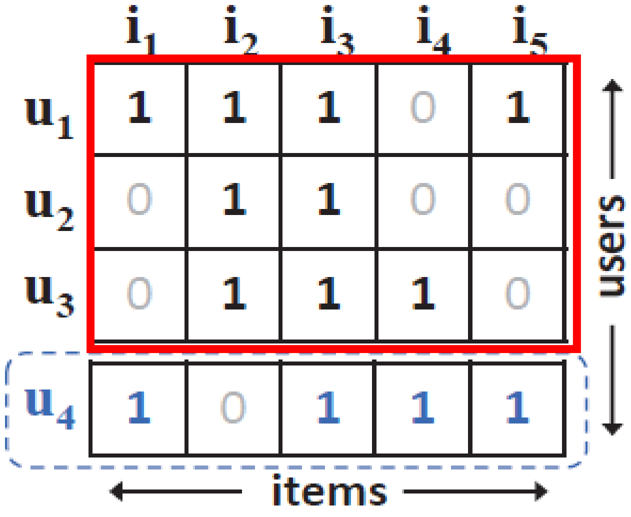
  - p_u and q_i denote latent vector for user u and item i
  - Linear model of latent factors

- Expressiveness Limitation of MF

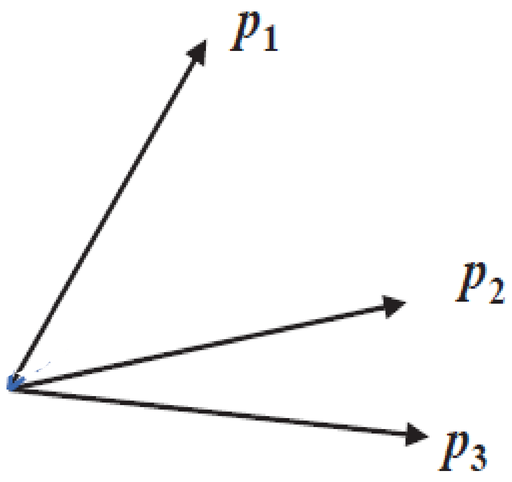
  1. Similarity between two users by inner product
     - Cosine similarity
  2. Use Jaccard coefficient
     - Ground-truth similarity of two users that MF needs to cover
     - Defined as $s_{ij} = \frac{|\mathcal{R}_i| \cap |\mathcal{R}_j|}{|\mathcal{R}_i| \cup |\mathcal{R}_j|}$, $\mathcal{R}_u$ be the set of items that user u has interacted with

Matrix Factorization

1. Consider only the first three users



(a) user–item matrix



(b) user latent space

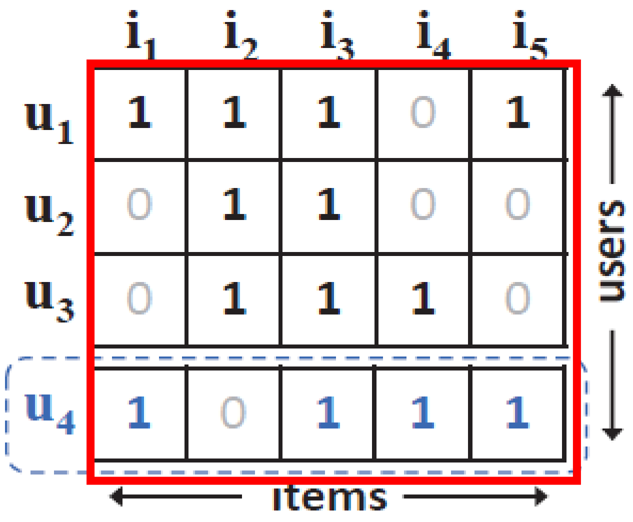
- Similarity

$$s_{23}\left(\frac{2}{3}=0.66\right) > s_{12}\left(\frac{2}{4}=0.5\right) > s_{13}\left(\frac{2}{5}=0.4\right)$$

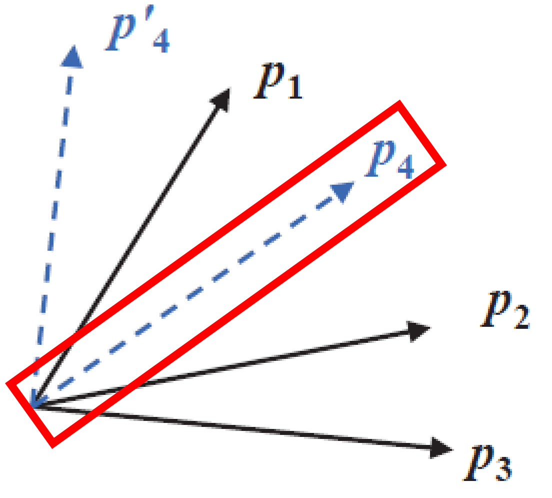
- Interpretation
  - Relative Similiarity

Matrix Factorization

2. Consider new user u_4



(a) user–item matrix



(b) user latent space

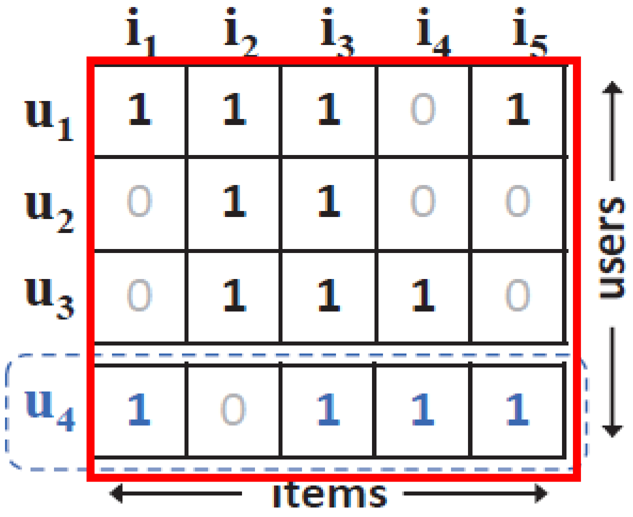
- Similarity

$$s_{41}\left(\frac{3}{5}=0.6\right) > s_{43}\left(\frac{2}{5}=0.4\right) > s_{42}\left(\frac{1}{5}=0.2\right)$$

- Interpretation
  - p_4 closer to p_2 than p_3

Matrix Factorization

Question : Increasing the number of latent factors K ?



(a) user–item matrix

Large Latent space

?

- It may hurt the generalization of the model (Overfitting)
  - Especially in sparse settings
- So use DNNs to learn the interaction function from data

General Framework

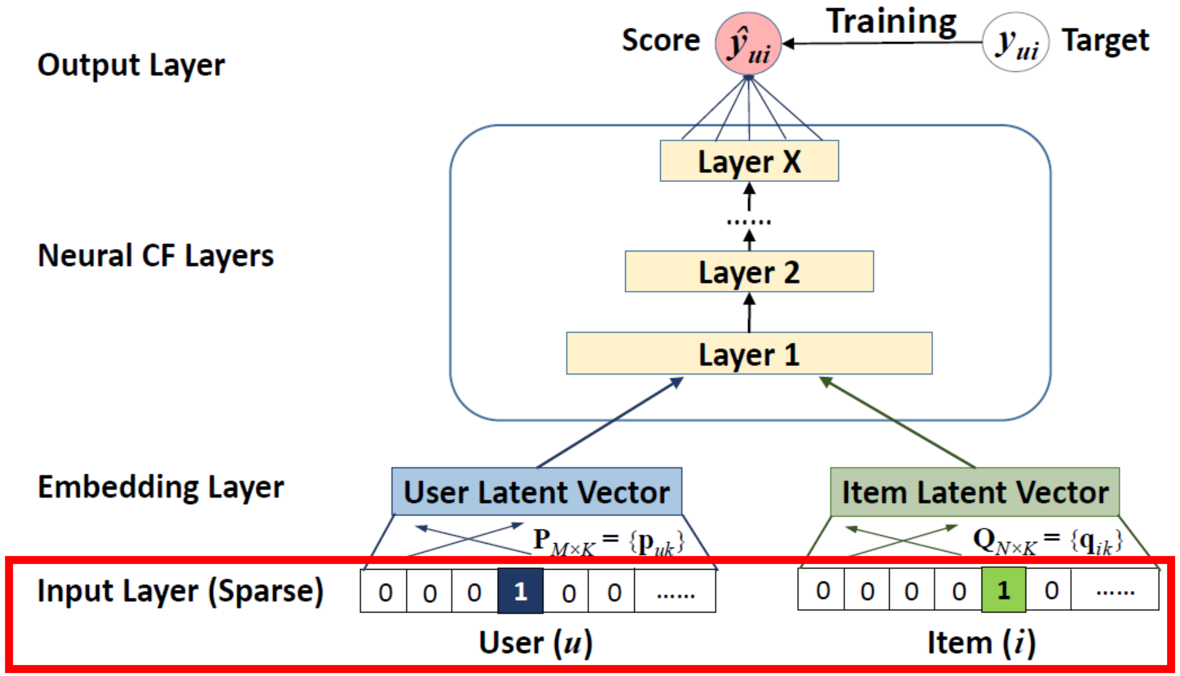


Figure 2: Neural collaborative filtering framework

- Multi-Layer Representation

1. Input Layer
    - Sparse feature vectors v_u^U and v_i^I
        - Binarized by one-hot encoding
        - Pure collaborative filtering setting

General Framework

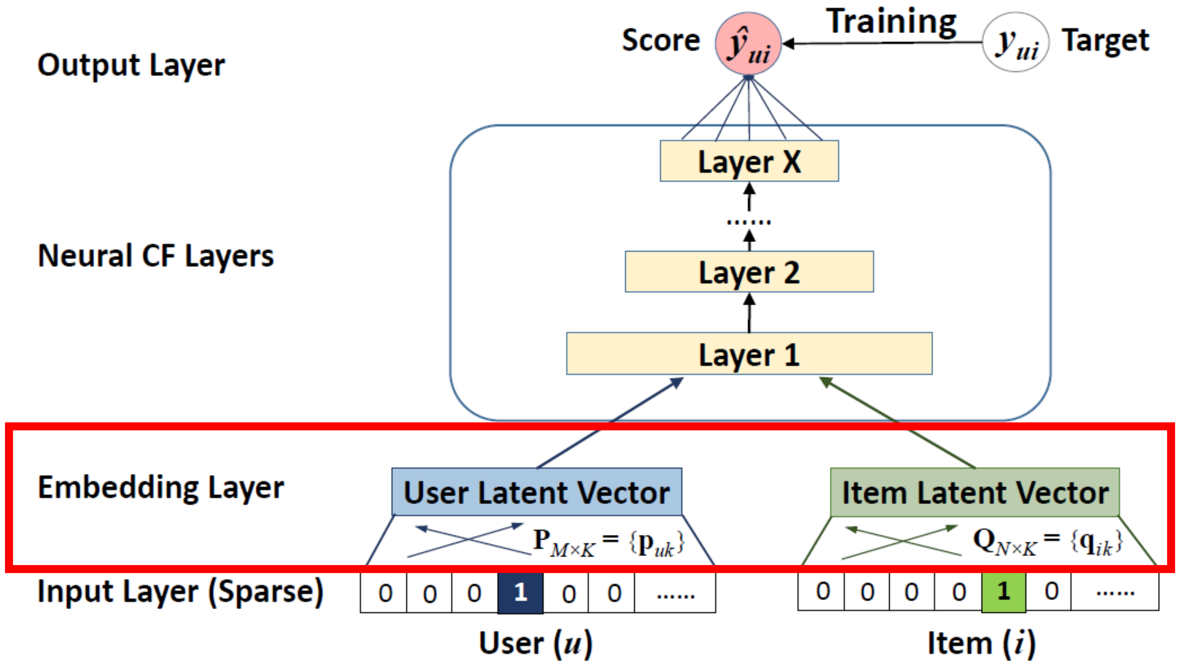


Figure 2: Neural collaborative filtering framework

2. Embedding Layer
   - Fully-Connected Layer
   - Sparse representation to a dense vector

General Framework

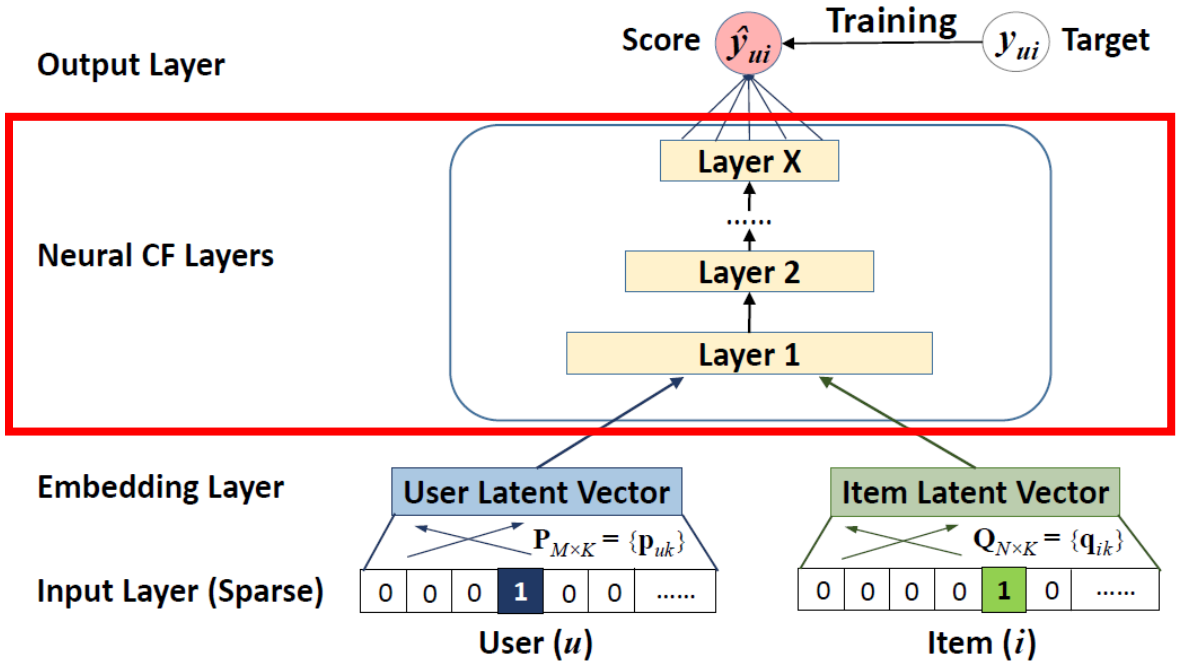


Figure 2: Neural collaborative filtering framework

3. Neural CF Layer
   - Multi-Layer Neural Architecture
   - Map the latent vectors to prediction scores
   - Dimension of last hidden Layer X determines the model's capability

General Framework

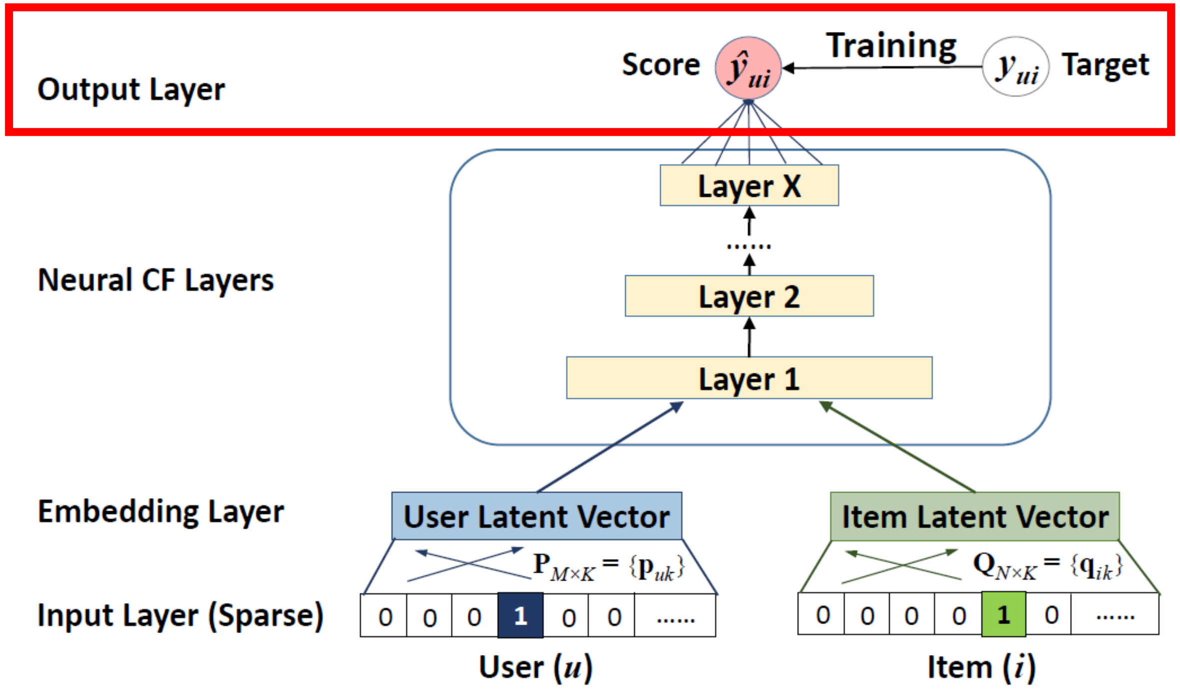


Figure 2: Neural collaborative filtering framework

4. Output Layer

- Predicted Score $\hat{y}_{ui}$
- By minimizing pointwise loss
  - ※ Future work for pairwise loss

- Abstraction

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

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

Learning NCF

- Existing Pointwise Methods

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

  - Squared Loss explained by Gaussian Distribution (Observation)
    - But target value $\mathrm{y_{ui}}$ is binarized

⬇

- Probabilistic Approach in NCF
  - $\mathrm{y_{ui}} = 1$ means item i is relevant to u, and 0 otherwise
  - $\hat{\mathrm{y}}_{\mathrm{ui}}$ represents how likely i is relevant to u
    - Use the Logistic activation Function to make $\hat{\mathrm{y}}_{\mathrm{ui}} \in [0,1]$

Learning NCF

- Define Likelihood Function

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

Negative Logarithm

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

- Same as Binary cross-entropy loss
  - Recommendation with implicit feedback as a binary classification problem
- Uniformly sample negative instances from unobserved interactions in each iteration

Generalized Matrix Factorization (GMF)

Let $\mathbf{p}_u$ be $\mathbf{P}^T\mathbf{v}_u^U$ and $\mathbf{q}_i$ be $\mathbf{Q}^T\mathbf{v}_i^I$

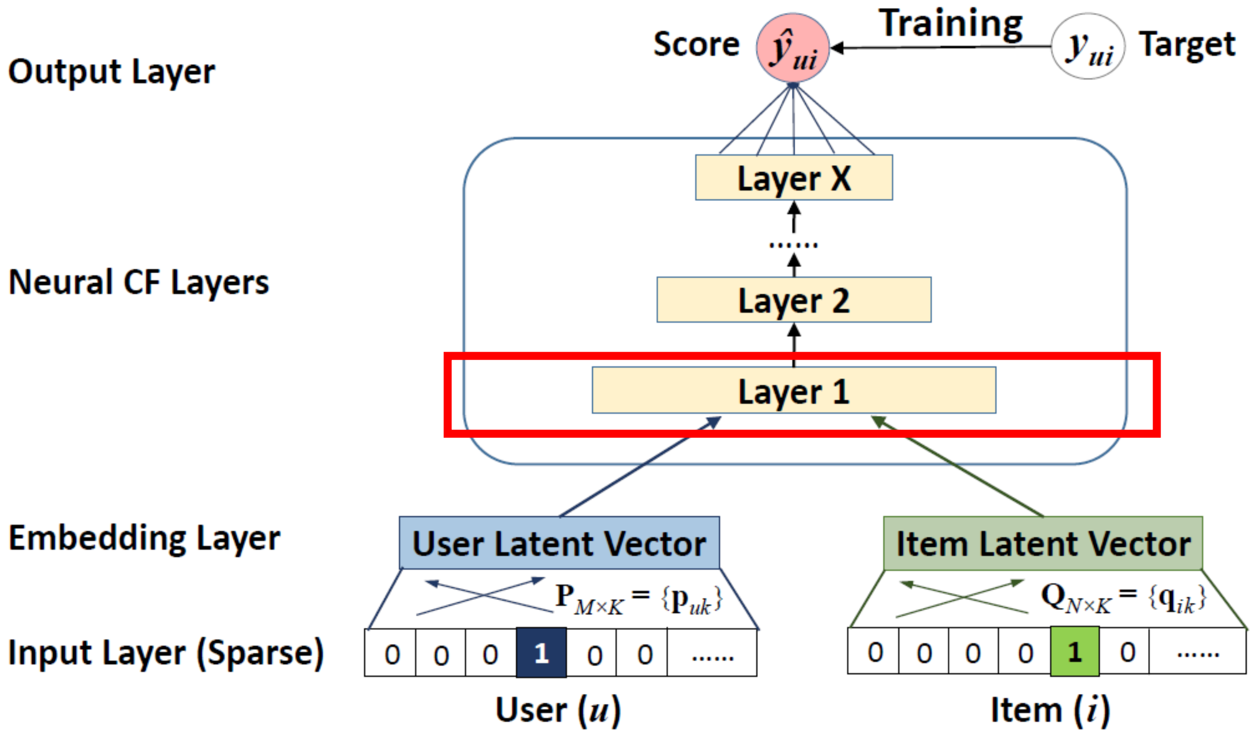


Figure 2: Neural collaborative filtering framework

- First CF Layer

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

$\odot$ denotes the element-wise product of vectors

Generalized Matrix Factorization (GMF)

Let $\mathbf{p}_u$ be $\mathbf{P}^T\mathbf{v}_u^U$ and $\mathbf{q}_i$ be $\mathbf{Q}^T\mathbf{v}_i^I$

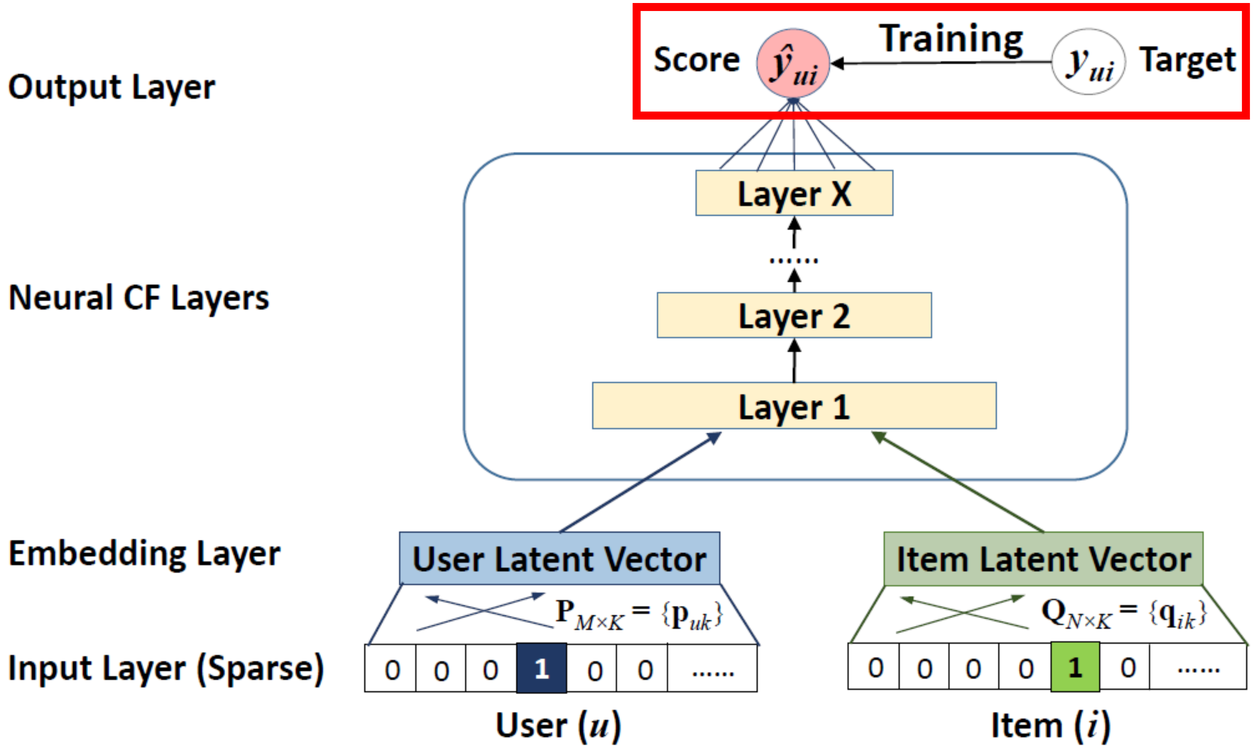


Figure 2: Neural collaborative filtering framework

- Output Layer

$$\hat{y}_{ui} = \underbrace{a_{out}}_{\text{Identity Function}}(\underbrace{\mathbf{h}^T}_{\text{Uniform Vector of 1}}(\mathbf{p}_u \odot \mathbf{q}_i))$$

Identity Function Uniform Vector of 1

"Become MF" $\hat{y}_{ui} = \mathbf{p}_u^T\mathbf{q}_i$

Generalized Matrix Factorization (GMF)

- Generalized and Extended Version of MF

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

1. Allow h to be learnt from Data
   - Vary importance of latent dimension
2. Use a non-linear function for a_{out}
   - Be more expressive than linear MF model

Multi-Layer Perceptron (MLP)

Let $\mathbf{p}_u$ be $\mathbf{P}^T\mathbf{v}_u^U$ and $\mathbf{q}_i$ be $\mathbf{Q}^T\mathbf{v}_i^I$

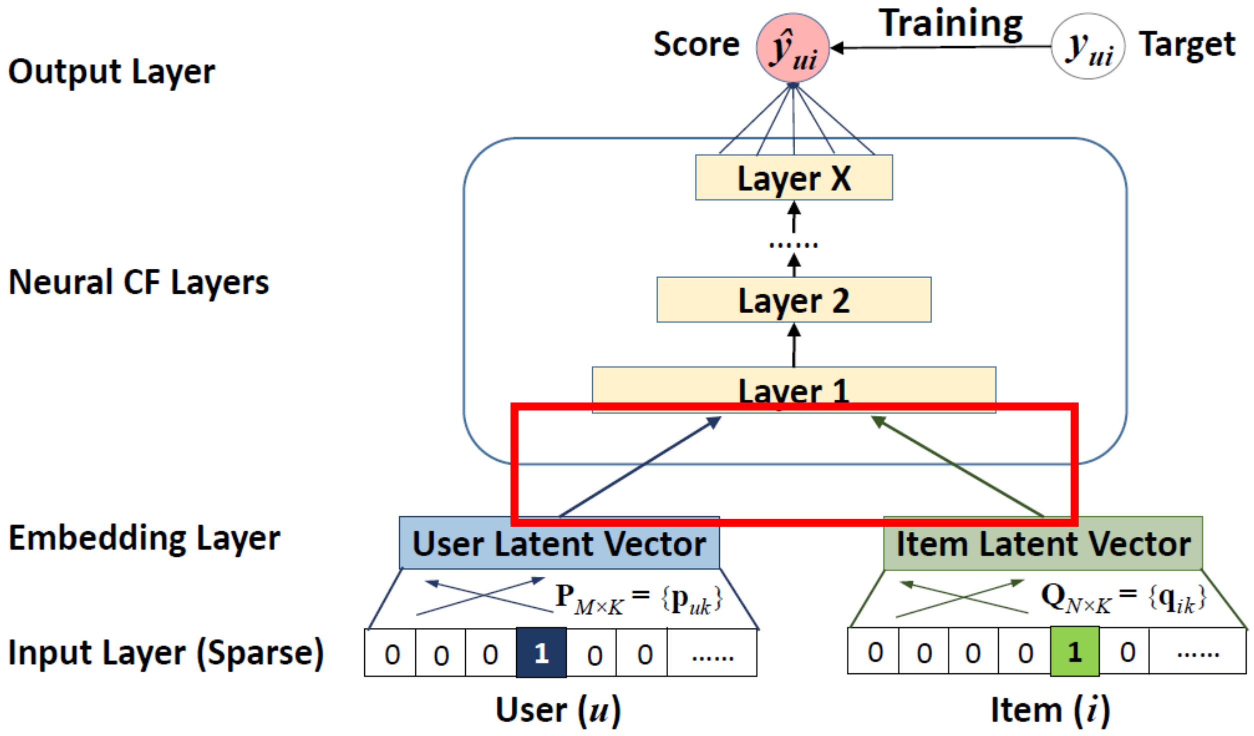


Figure 2: Neural collaborative filtering framework

- Hidden Layers on the concatenated vector
- Layer Composition

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

- ReLU yields slightly better

Fusion of GMF and MLP

✓ GMF uses a linear kernel & MLP uses a non-linear kernel

⬇ Natural Question

? How can we fuse GMF and MLP, so that they make a better model?

⬇

1. Share same embedding layer and

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

   - From Neural Tensor Network
   - Limit the performance
     - Same size of embeddings

➡ ✓ Separate embedding layer and then combine

Fusion of GMF and MLP

$\mathbf{p}_u^G$ and $\mathbf{p}_u^M$ denote user embeddings, $\mathbf{q}_i^M$ and $\mathbf{q}_i^G$ denote item embeddings

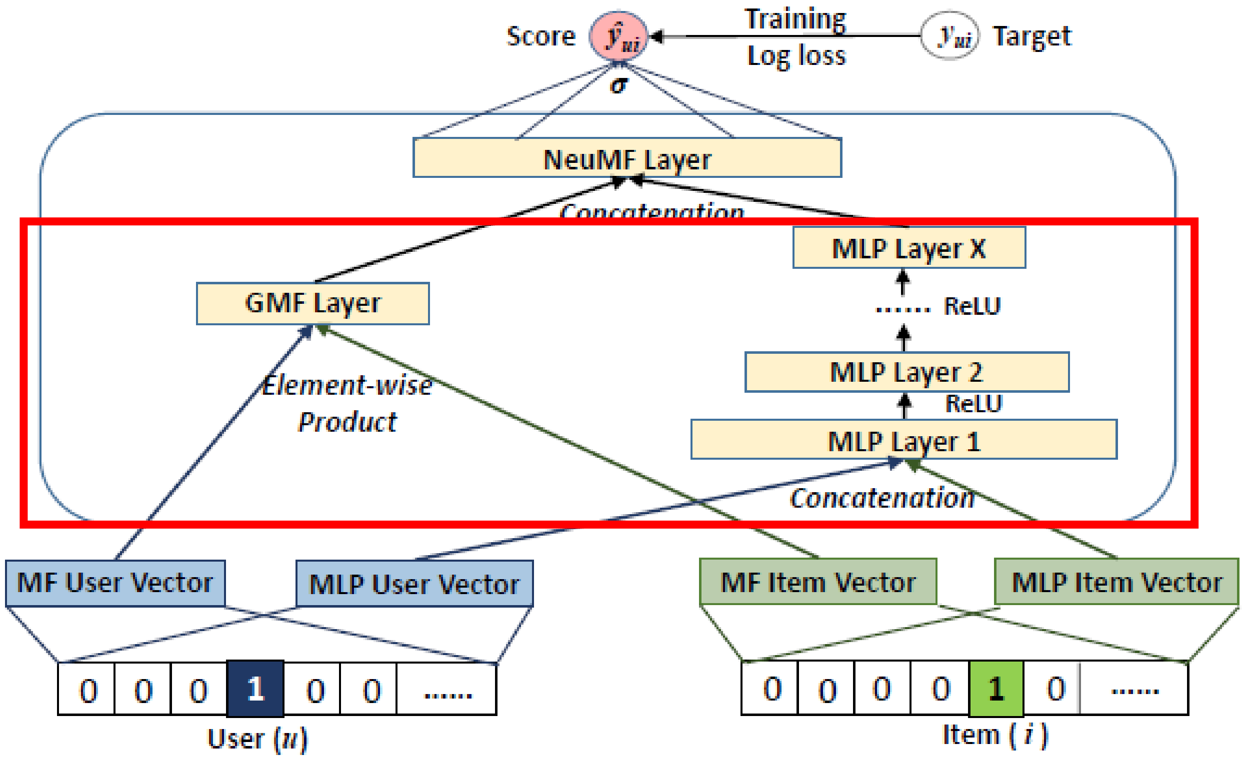


Figure 3: Neural matrix factorization model

- Combines linearity of MF and non-linearity of DNNs
- Layer Composition

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

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

Fusion of GMF and MLP

$\mathbf{p}_u^G$ and $\mathbf{p}_u^M$ denote user embeddings, $\mathbf{q}_i^M$ and $\mathbf{q}_i^G$ denote item embeddings

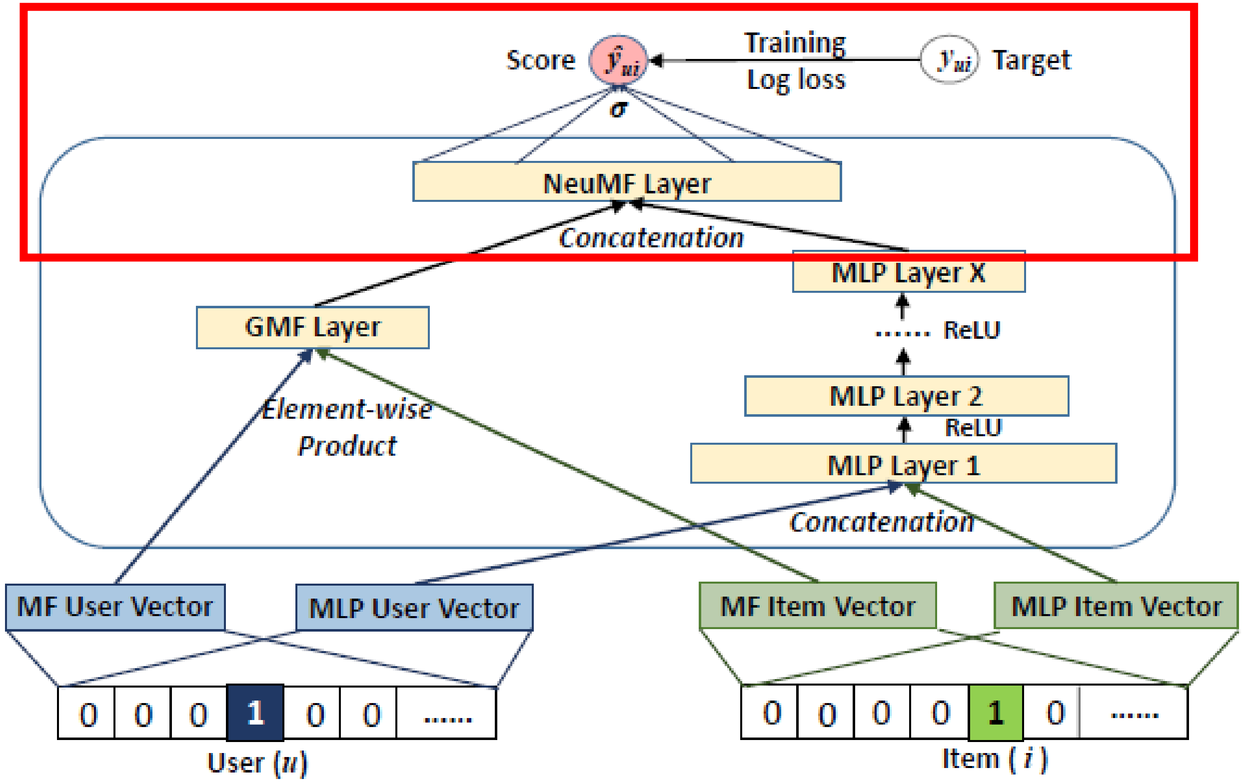


Figure 3: Neural matrix factorization model

- Hidden Layers on the concatenated vector
- Layer Composition

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

"Neural Matrix Factorization"

Pre-Training

- Non-convexity of the objective function of NeuMF

⬇

- Initialization determines convergence and performance

⬇

- Initialize NeuMF using the pretrained models of GMF and MLP
  1. Train GMF and MLP with random initializations
  2. Use their model parameters as the initialization

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

- Weighted concatenation of pretrained $\mathbf{h}$ vector

Research Questions

- Conduct experiments with the aim of answering three questions

1. Do NCF methods outperform?
2. How does optimization framework work for the recommendation task?
3. Are deeper layers of hidden units helpful for learning from data?

Experimental Settings

I. Datasets

1. MovieLens

- Movie rating dataset
  - Learning from implicit signal of explicit feedback
  - Whether user has rated item

2. Pinterest

- SNS for sharing images
  - Constructed by below paper
  - X. Geng, H. Zhang, J. Bian, and T.-S. Chua. Learning image and user features for recommendation in social networks. In ICCV, pages 4274-4282, 2015.

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%

Experimental Settings

II. Evaluations Protocols

- Leave-one-out (latest interaction as test, remaining for train)
- Rank the test item among 100 unobserved randomly sampled items

1. HR (Hit Ratio)

- Whether test item is present on the top-K list

2. NDCG

- The position of the hit by assigning higher scores to hits at top ranks

III. Baselines

1. ItemPop

- Non-personalized

2. ItemKNN

- Item-based CF

3. BPR

- Pairwise Loss

4. eALS

- State-of-the-art MF

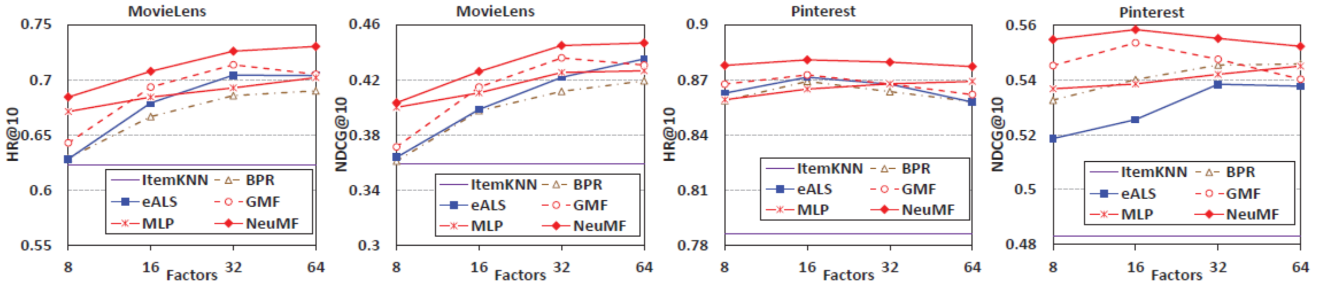
Experimental Settings

IV. Parameter Settings

- To determine hyperparameters, randomly sampled one interaction for each user as the validation data
- Initialized model parameters with a Gaussian Distribution – $N(0, 0.01^2)$
- Optimizing with mini-batch Adam (batch size – [128,256,512,1024])
- Learning rate – [0.0001, 0.0005, 0.001, 0.005]
- Last hidden layer determines the model capability - term it as predictive factors [8, 16, 32, 64]
  - Ex) Architecture of NCF layers 32 -> 16 (embedding size) -> 8, three hidden layers
- Set alpha = 0.5 for pre-training

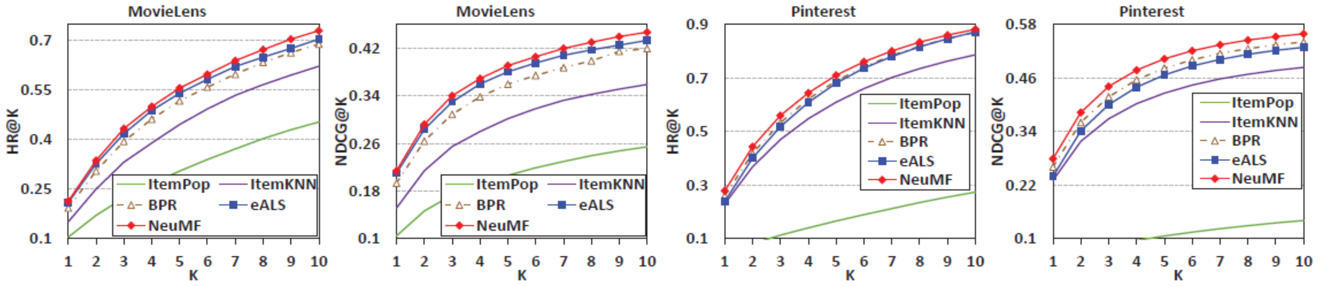
Performance Comparison (RQ1)

- Performance 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.

④ NeuMF achieves the best performance on both datasets

④ GMF and MLP also show quite strong performance

④ GMF shows consistent improvements over BPR

⑤ NeuMF demonstrates consistent improvements

Performance Comparison (RQ1)

- Utility of Pre-training

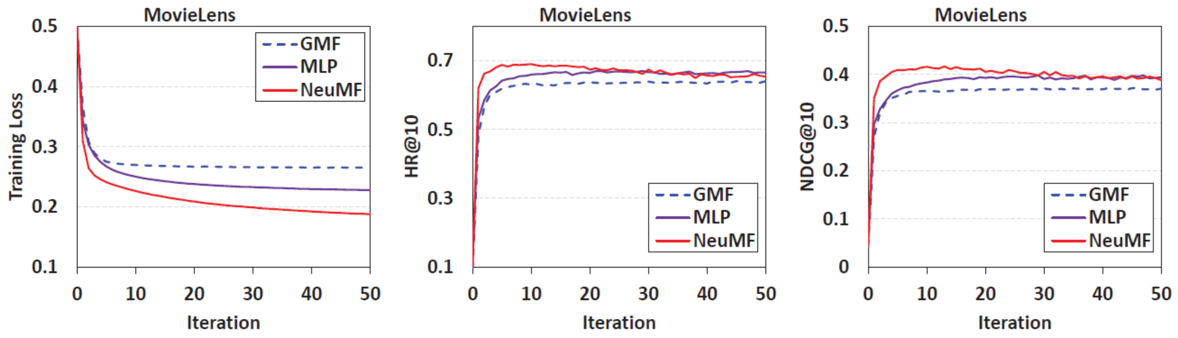
Table 2: Performance of NeuMF with and without pre-training.

	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

- NeuMF with pre-training achieves better performance in most cases

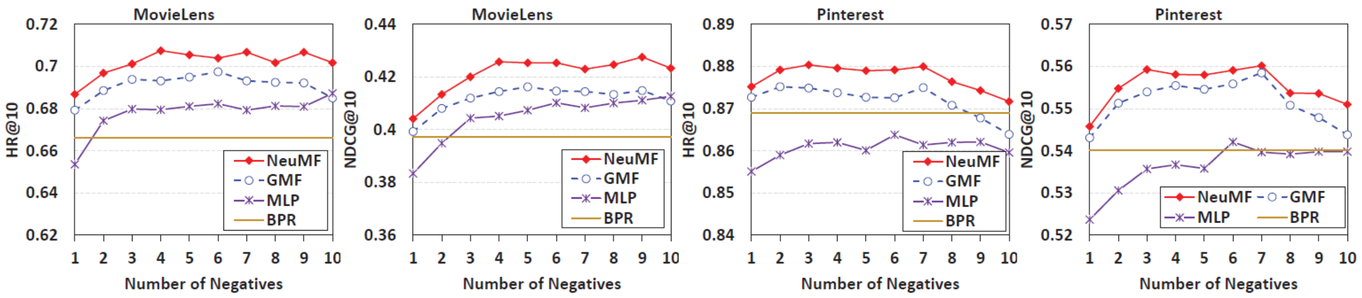
Log Loss with Negative Sampling (RQ2)

- Result of Pointwise Log loss



(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.

⑥ With more iterations, loss gradually decreases

⑥ NeuMF achieves the lowest loss followed by MLP, GMF

⑦ Pointwise loss can handle flexible sampling ratio

⑦ Too large sampling ratio hurts the performance

Is Deep Learning Helpful? (RQ3)

- MLP with different layers

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**

- Stacking more layers are beneficial to performance
  - High non-linearities brought by stacking more non-linear layers
- Simply concatenating latent vectors is insufficient for modelling (MLP with no hidden layers)

Conclusion and Future Work

- Explored Neural Network Architectures for CF
  - Simple and general framework NCF
    - Three instantiations – GMF, MLP, NeuMF
    - Opening up a new avenue for recommendation based on Deep Learning

- Future Work
  - Study pairwise learners for NCF
  - Extend NCF to model auxiliary information
    - Such as user review, knowledge bases, temporal signals
  - Build recommender systems for multi-media items
    - Effective methods for multi-view and multi-modal data

Python Implementation

- To be updated (Not complete yet)
  - Having trouble dealing with argparse and non-csv file
  - https://github.com/LeeJunmo/DSAIL-Lab-Intern

Any Questions?