MIE524 Data Mining

Recommender Systems:
Neural Collaborative Filtering

Some slides modified from He Xiangnan

Reminder: Collaborative Filtering

- **Does not require profiles (features) for items or users**
  - Different from content-based recommender systems

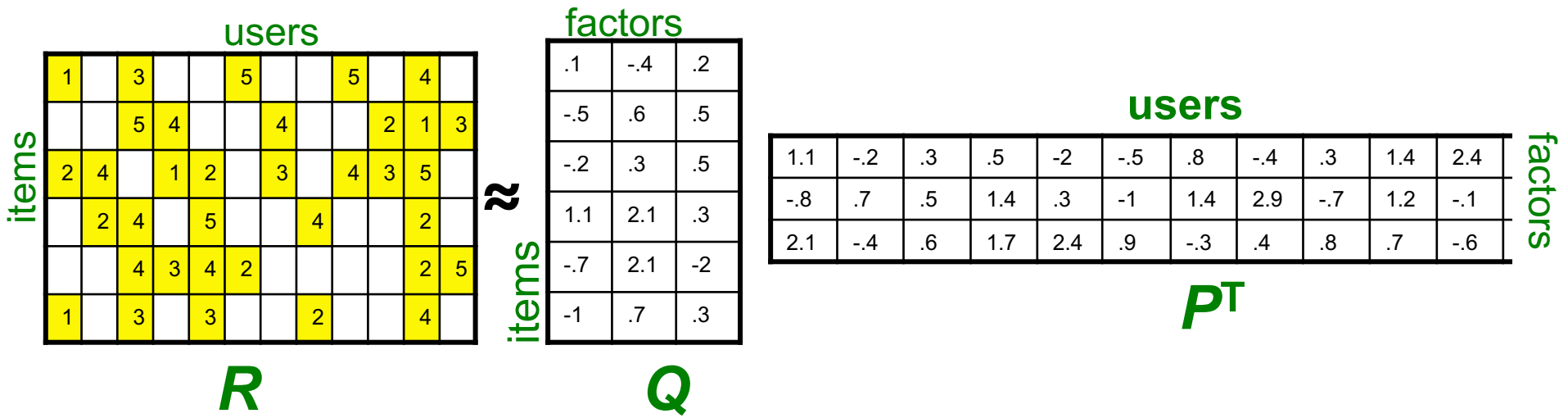
- **Instead, only requires user-item interaction data**
  - ***Explicit*** feedback
    - e.g., 0-5 stars
  - ***Implicit*** feedback
    - e.g., user watched movie (0/1)

Reminder: Latent Factor Models

- Approximate the rating matrix **R** as product of matrices **Q** · **P**^{T}
  - **Problem**: **R** has missing entries
  - Learn Q and P that minimize the **reconstruction error** on known ratings and ignore the missing values

$$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$$

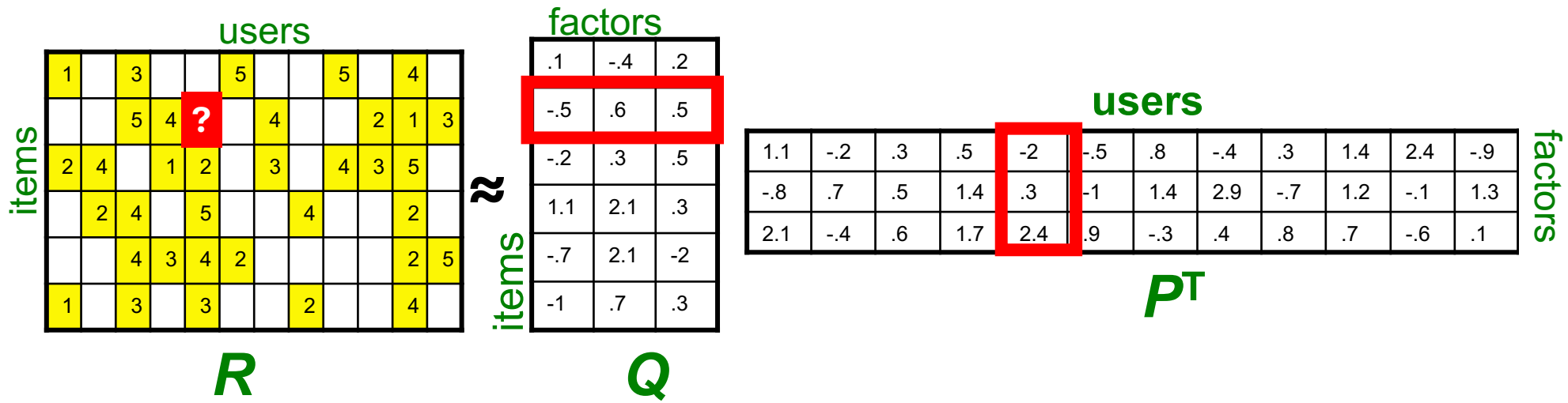
q_i = row i of Q
p_x = column x of P^T



Reminder: Latent Factor Models

- **How to estimate missing ratings?**
- Estimate using latent factors $\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$

q_i = row i of Q
p_x = column x of P^T



Motivation for Neural Approaches

- **Matrix Factorization is restricted by the simple choice of interaction function between users and items -- the dot (inner) product**
  - Inner product restricted to simple linear combination of multiplication of latent features
  - may not be sufficient to capture the complex structure of user interaction data
  - We've seen that even simple incorporation of user/item biases into the interaction function can improve performance

- **Can we use (deep) neural networks to learn the interaction function from data?**

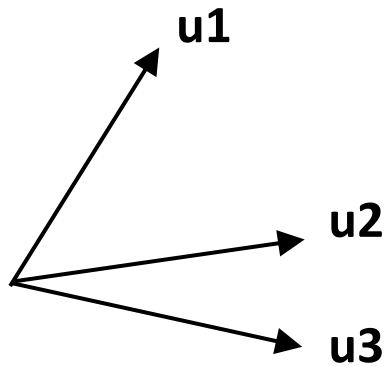
Motivating Example

$$\hat{y}_{ui} = \mathbf{U}_i^T \mathbf{V}_j \simeq cos(\mathbf{U}_i, \mathbf{V}_j)$$ (E.g., assuming a unit length)

	$\mathbf{i}_1$	$\mathbf{i}_2$	$\mathbf{i}_3$	$\mathbf{i}_4$	$\mathbf{i}_5$
$\mathbf{u}_1$	1	1	1	0	1
$\mathbf{u}_2$	0	1	1	0	0
$\mathbf{u}_3$	0	1	1	1	0

$$sim(u_i, u_j) = \frac{|u_i \cap u_j|}{|u_i \cup u_j|}$$

sim(u1,u2) = 0.5
sim(u3,u1) = 0.4
sim(u3,u2) = 0.66



2-d embedding

Motivating Example

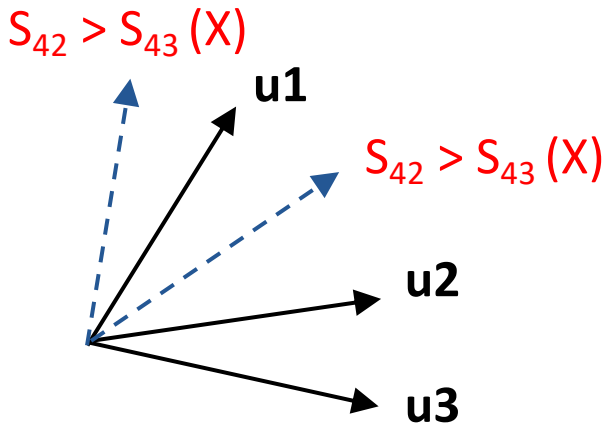
$$\hat{y}_{ui} = \mathbf{U}_i^T \mathbf{V}_j \simeq cos(\mathbf{U}_i, \mathbf{V}_j)$$ (E.g., assuming a unit length)

	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

$$sim(u_i, u_j) = \frac{|u_i \cap u_j|}{|u_i \cup u_j|}$$

sim(u1,u2) = 0.5
sim(u3,u1) = 0.4
sim(u3,u2) = 0.66

sim(u4,u1) = 0.6
sim(u4,u2) = 0.2
sim(u4, u3) = 0.4



2-d embedding

Motivating Example

$$\hat{y}_{ui} = \mathbf{U}_i^T \mathbf{V}_j \simeq cos(\mathbf{U}_i, \mathbf{V}_j)$$ (E.g., assuming a unit length)

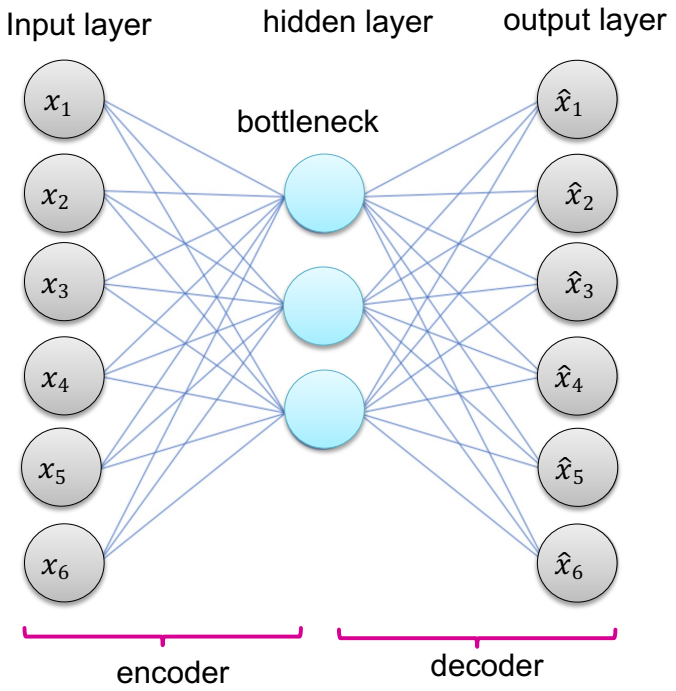
How to resolve?

- **Increase the number of latent dimensions**
  - **May lead to overfitting in sparse environment**

- **Replacing the simple, fixed inner product with interaction function learned from the data**
  - **AutoRec: Autoencoders Meet Collaborative Filtering**
  - **NCF: Neural Collaborative Filtering**

Autoencoders + CF = AutoRec

- **Autoencoders (recall lecture 3)**
  - Learning concise representation by training to reconstruct input
    - Representations can be used for downstream tasks
  - Can be used to denoise input data
    - Reconstruct data from noised input
    - input: data + noise, output: data
  - **In recommender systems:**
    - **Can we learn to 'reconstruct' missing rating data?**

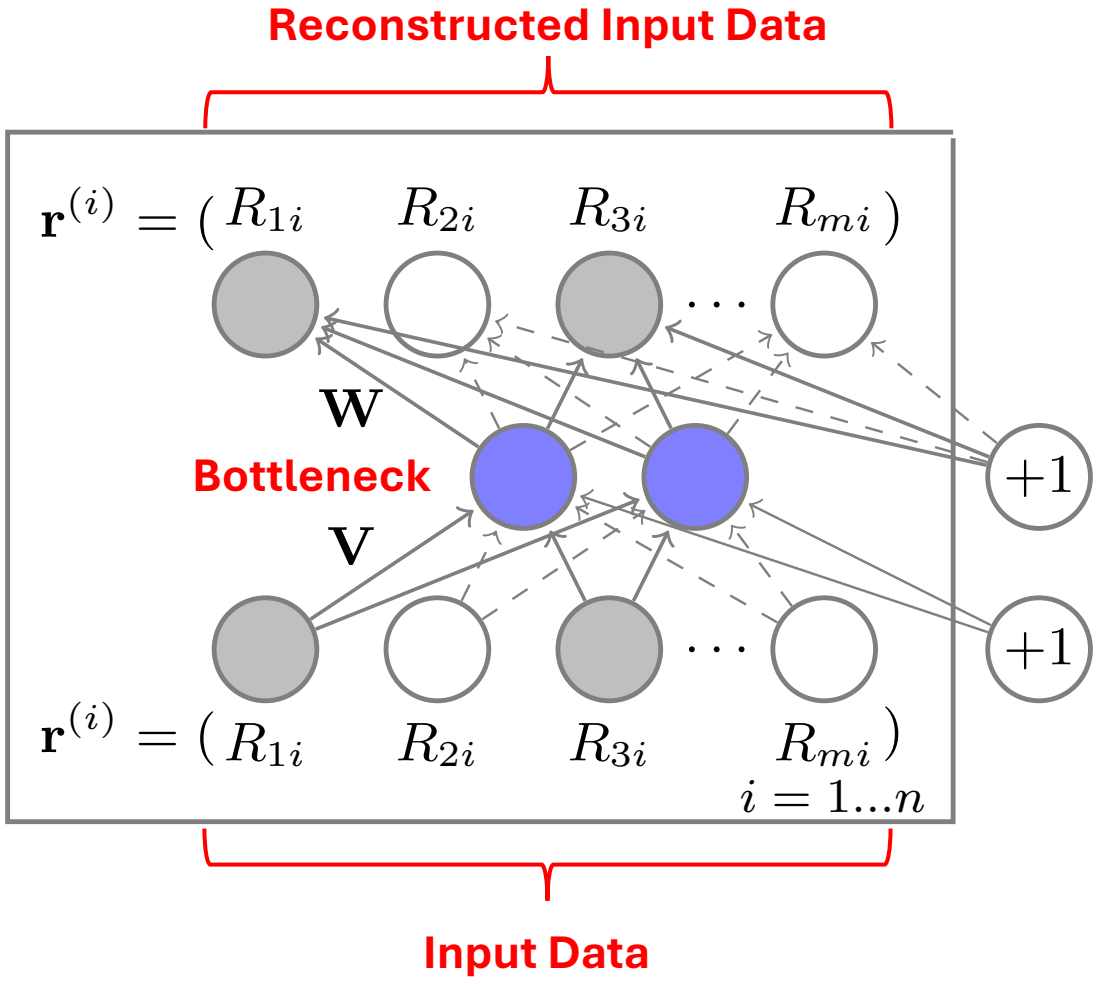


Autoencoders + CF = AutoRec

users

items

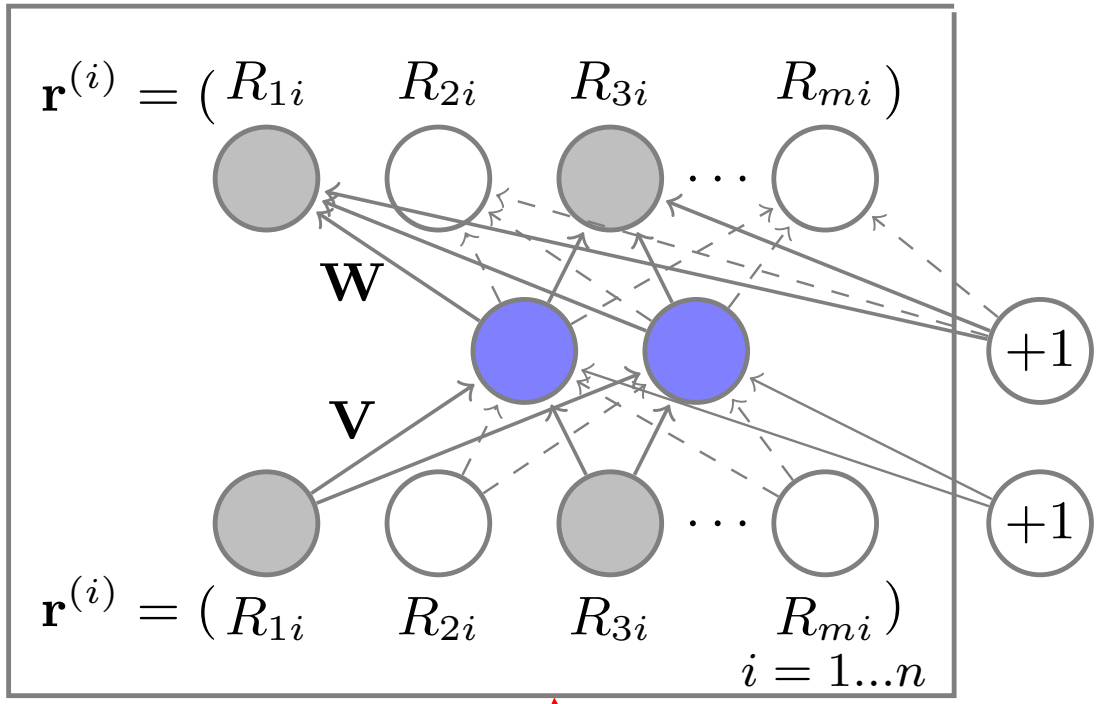
1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	



[Sedhain, S., Menon, A. K., Sanner, S., & Xie, L., 2015]

AutoRec: Training on existing ratings

$$\min_{\theta} \sum_{i=1}^{n} \underbrace{||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta))||_{\mathcal{O}}^2}_{\text{Reconstruct existing ratings}} + \underbrace{\frac{\lambda}{2} \cdot (||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2)}_{\text{Regularization}}$$



users

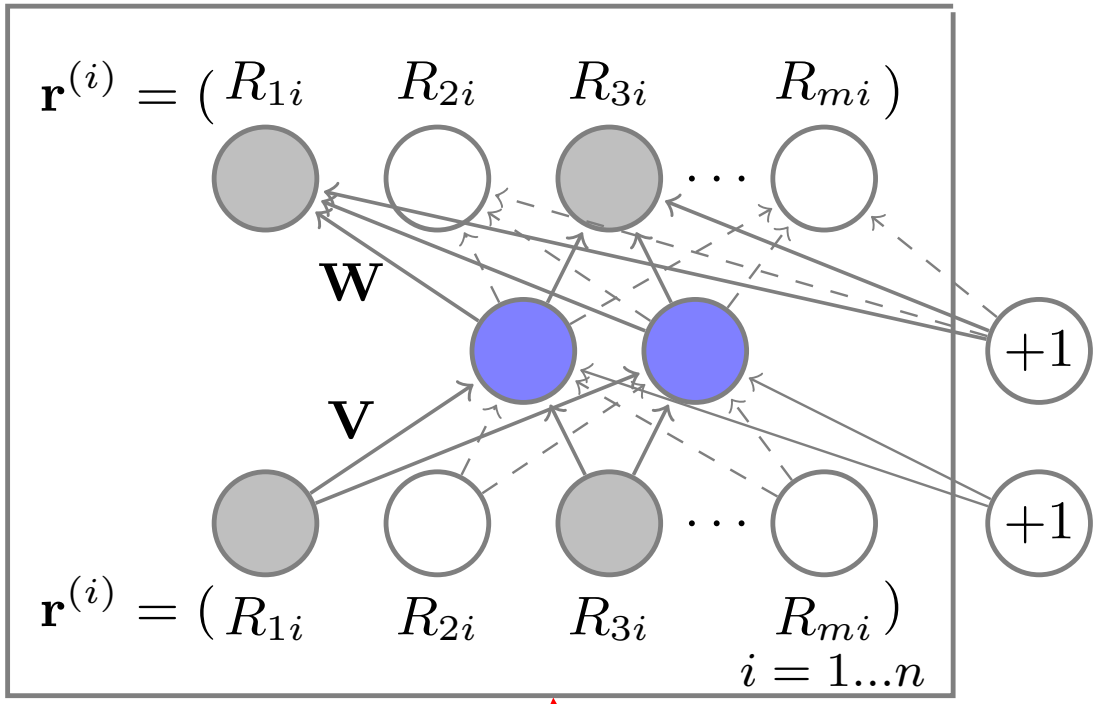
items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

[Sedhain, S., Menon, A. K., Sanner, S., & Xie, L., 2015]

AutoRec: Predicting missing ratings

$$\hat{R}_{ui} = (h(\mathbf{r}^{(i)}; \hat{\theta}))_u$$



users

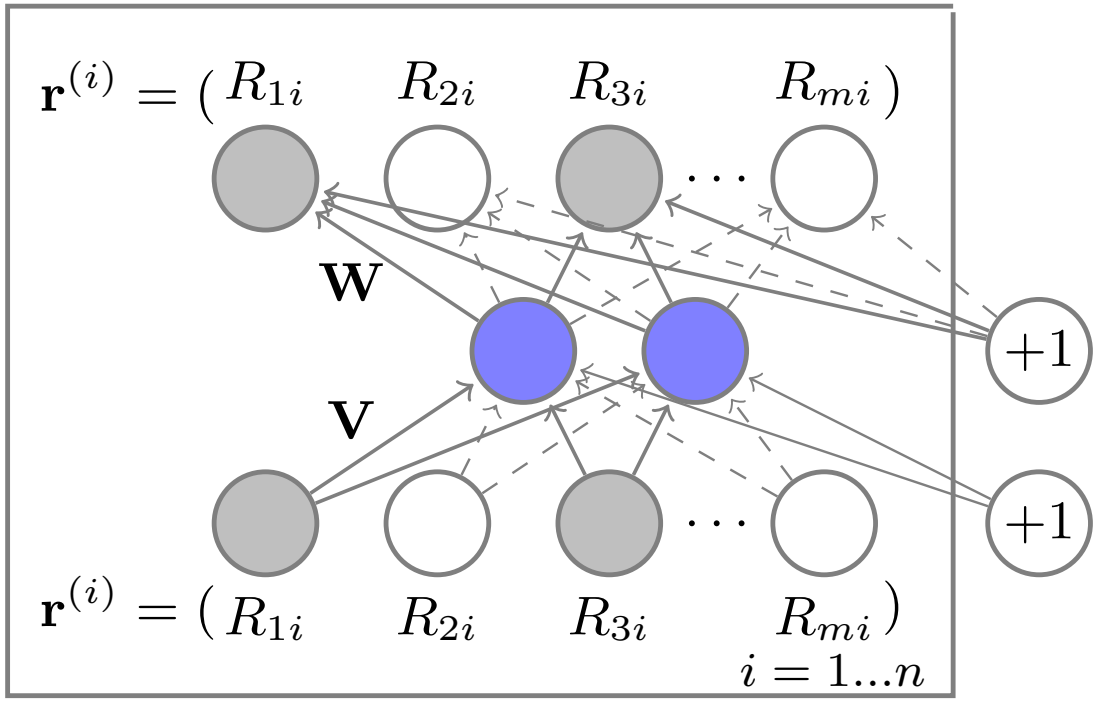
items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2		?			2	5
1		3		3			2			4	

[Sedhain, S., Menon, A. K., Sanner, S., & Xie, L., 2015]

AutoRec: Predicting missing ratings

$$\hat{R}_{ui} = (h(\mathbf{r}^{(i)}; \hat{\theta}))_u$$



$$h(\mathbf{r}; \theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

Choice of linear and non-linear activation functions:

$f(\cdot)$	$g(\cdot)$	RMSE
Identity	Identity	0.872
Sigmoid	Identity	0.852
Identity	Sigmoid	**0.831**
Sigmoid	Sigmoid	0.836

[Sedhain, S., Menon, A. K., Sanner, S., & Xie, L., 2015]

Neural Collaborative Filtering

- Three approaches:

- **Generalized Matrix Factorization (GMF)**
  - a simple neural network that generalize and extend matrix factorization

- **Multi-layer perceptron (MLP)**
  - a deep neural network that incorporates nonlinearities in an MLP architecture

- **Neural Collaborative Filtering (NCF)**
  - a framework that combines both GMF and MLP

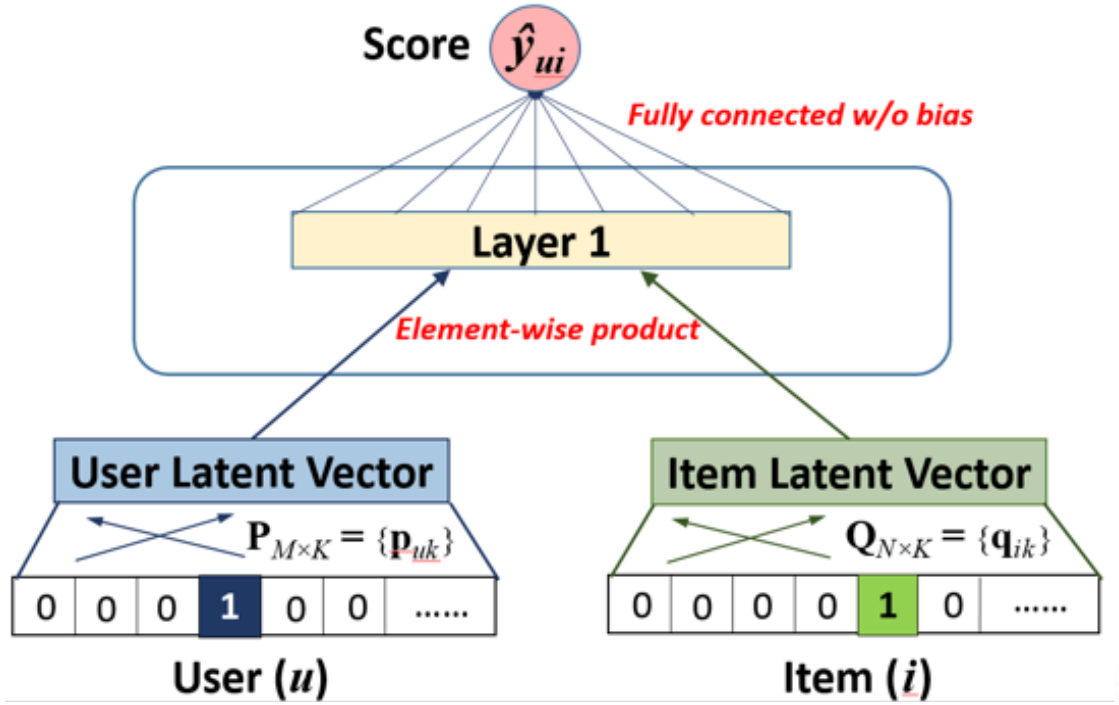
[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Generalized Matrix Factorization (GMF)

- K-dimensional "Shallow" (lookup) embedding for each user and each item (similar to Funk's SVD).

- Layer 1: element-wise product
- Output layer: fully-connected w/o bias

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



[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

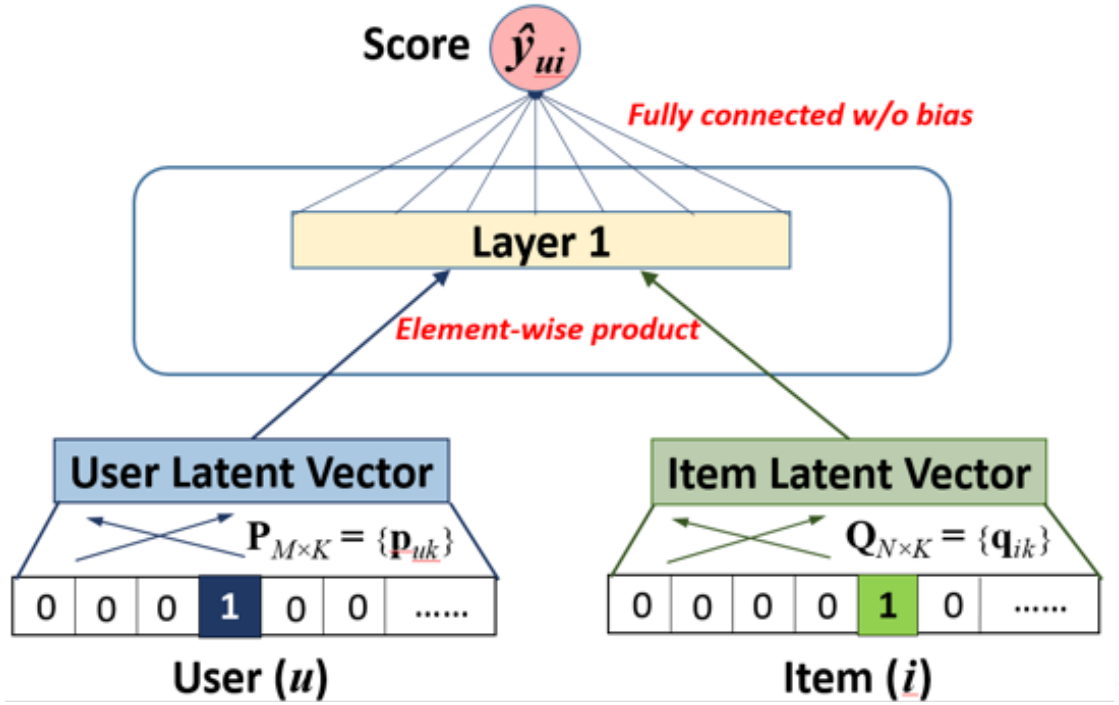
Generalized Matrix Factorization (GMF)

- K-dimensional "Shallow" (lookup) embedding for each user and each item (similar to Funk's SVD).

- Layer 1: element-wise product
- Output layer: fully-connected w/o bias

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

Matrix Factorization (e.g., in Funk's SVD) if h=[1,..,1]



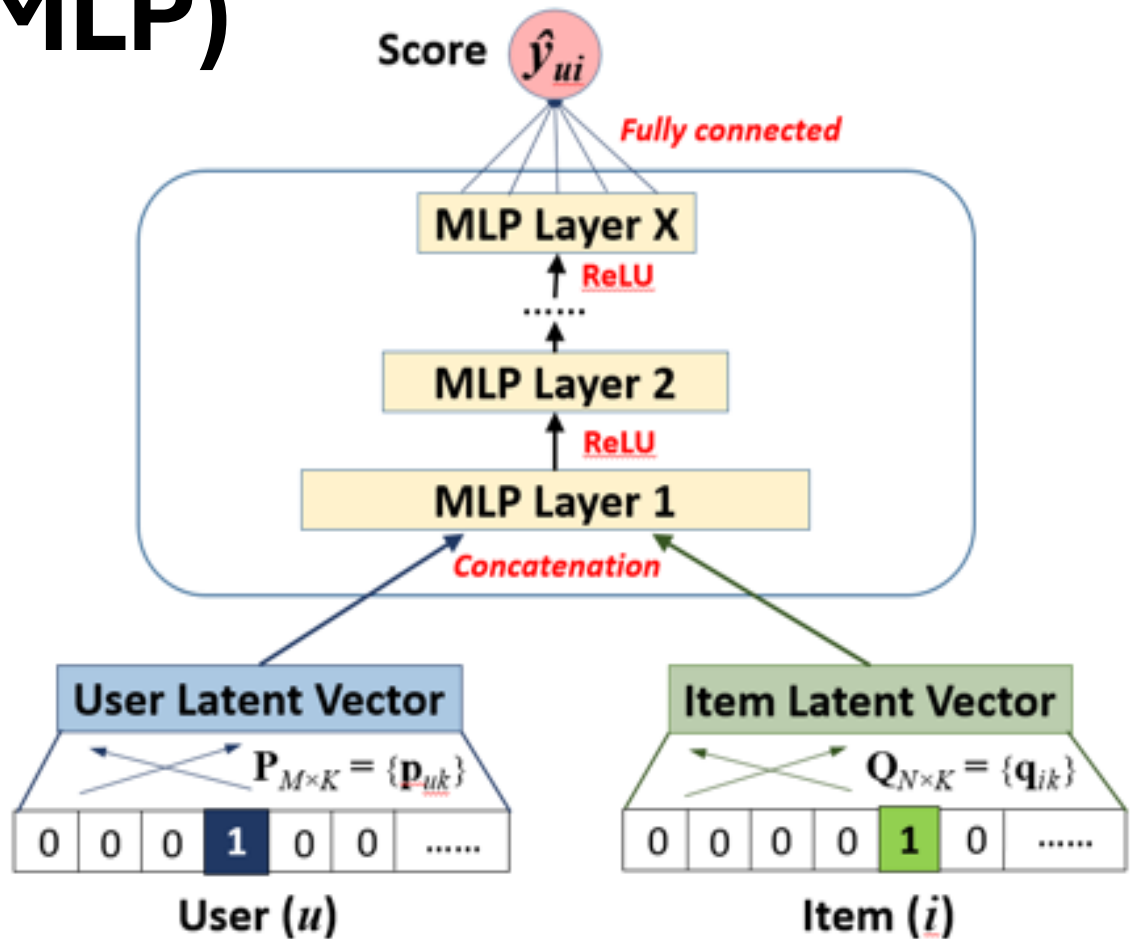
[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Multi-Layer Perceptron (MLP)

- K-dimensional "Shallow" (lookup) embedding for each user and each item (similar to Funk's SVD).

- Layer 1: concatenate user-item embeddings

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



- Remaining layers: fully-connected w/ ReLU

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

(G)MF vs. MLP

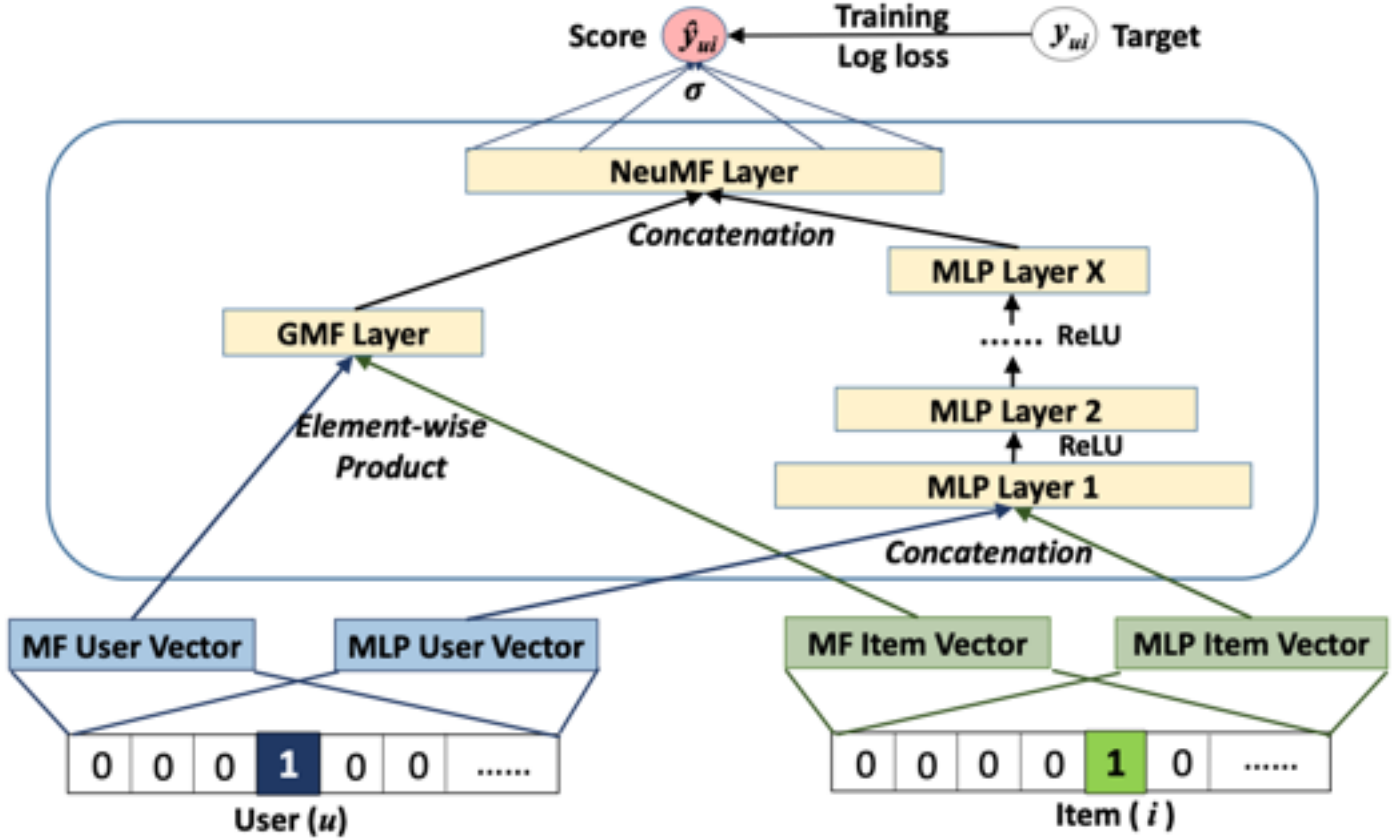
- **MF uses dot (inner) product as the interaction function**
  - Latent factors are independent of each other
  - Empirically, has good generalization ability for CF
- **MLP uses nonlinear functions to learn interaction function**
  - Latent factors are not independent of each other
  - The interaction is learned from data, which should have better representation ability
  - Risk of overfitting
- **Can we combine the two models together?**

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Neural Collaborative Filtering (NCF)

- NCF combines GMF and MLP by allowing them to learn different sets of embeddings

- **Output:**
  - **Concatenate GMF and last MLP layer**
  - **Fully-connected layer to output score**



[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Training NCF Models

- **For explicit feedback (e.g., ratings on 1-5 scale)**
  - Use regression loss:

$$L_r = \sum_{(u,i)\in\mathcal{Y}} \underbrace{(y_{ui} - \hat{y}_{ui})^2}_{\text{Accuracy}} + \underbrace{\lambda_\Theta ||\Theta||^2}_{\text{Regularization}}$$

- **For implicit feedback (e.g., watched video 0/1)**
  - Use binary classification loss

- **Train using SGD-based algorithm (e.g., Adam)**

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Training NCF Models

- “Tricks” and Enhancements:
  - **Pre-training:**
    - Train GMF model and MLP model on their own
    - Then initialize the weights of the combined NCF model using the weights learned independently for GMF and MLP
  - **Negative sampling for implicit feedback:**
    - Implicit feedback is trained as a binary classification
    - Typically, the large majority of interactions are (implied) negatives
    - Instead, uniformly sample negatives while controlling ratio of negatives to positives

NCF Performance

- **Two datasets from MovieLens and Pinterest:**
  - Focus on implicit feedback 0/1 (e.g., will the user watch a movie)

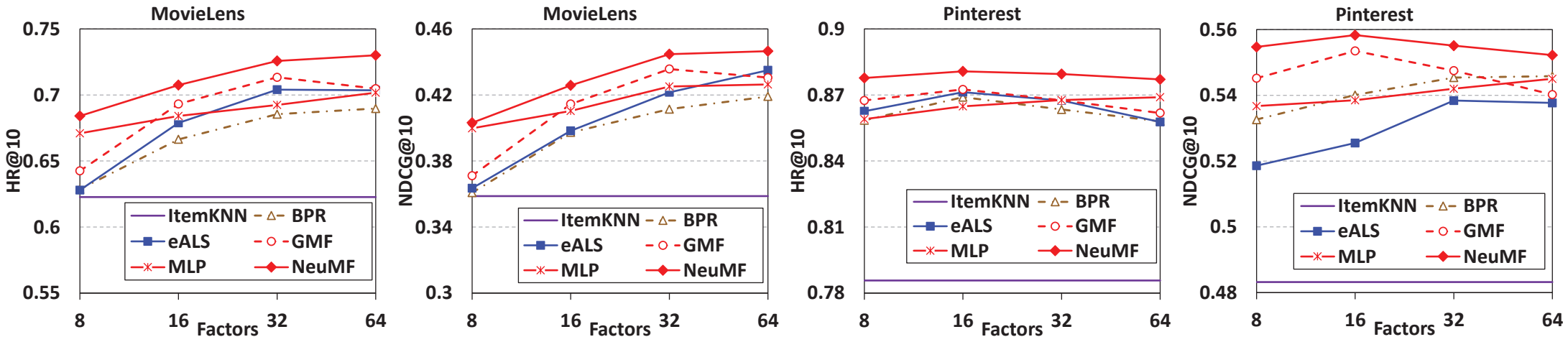
- **Evaluation:**
  - Leave-one-out: hold out latest rating for each user as test set

  - **Hit Ratio @10**
    - Does the test item appears in the top 10 recommendations for user?

  - **NDCG @10**
    - account for the position of “hit”: higher score for to hits at top ranks

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

NCF Performance



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

- NCF outperforms the baselines
- NCF > GMF > MLP (why GMF outperforms MLP?)

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

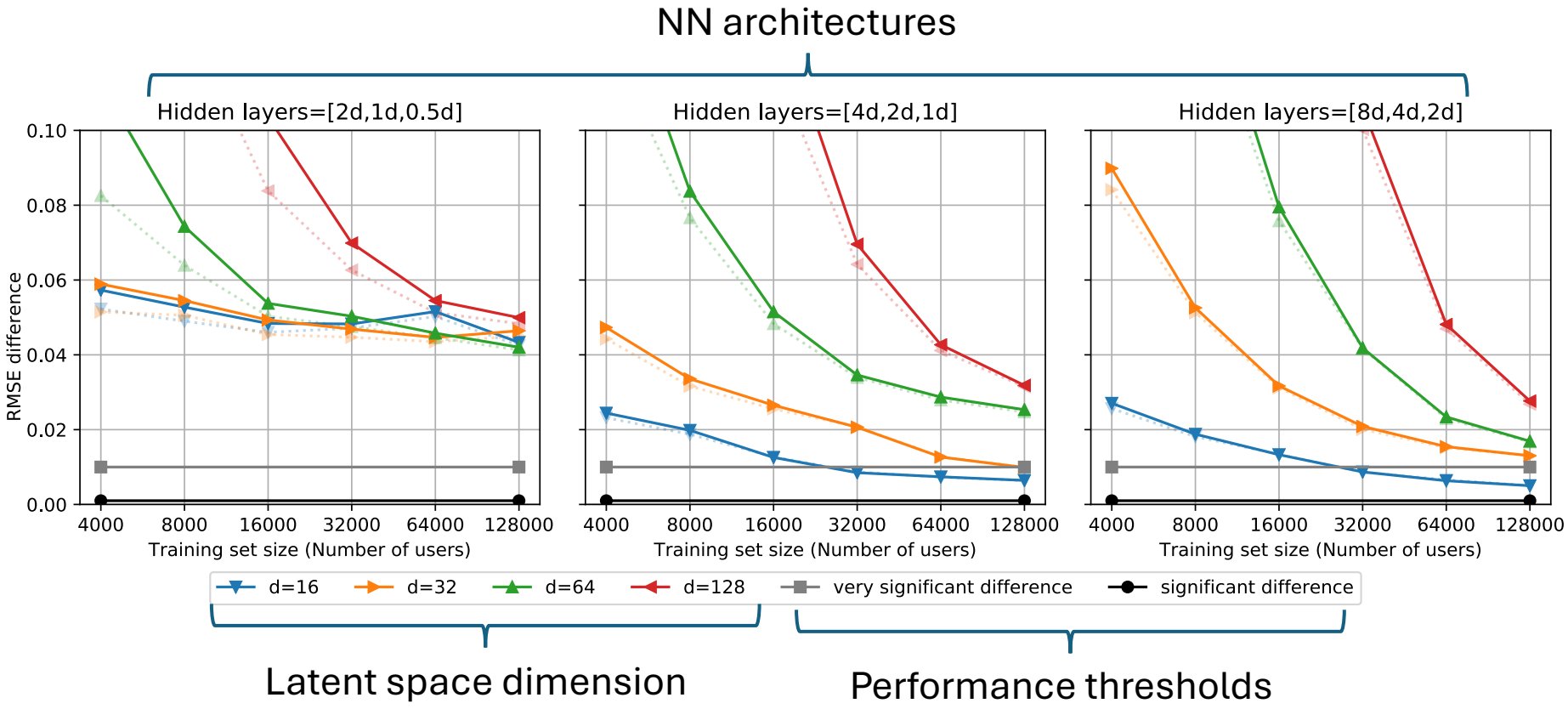
Why GMF alone outperforms MLP alone?

- #1: Overfitting
  - MLP learns more expressive interaction functions
  - Susceptible to fit noise
  - Generalize more poorly than MF

- #2: Learning dot product is hard
  - Dot product is empirically a good interaction function
  - In theory, MLP can be approximate any function. In practice, however, it is non-trivial to learn a dot product.

[Rendle, S., Krichene, W., Zhang, L., & Anderson, J., 2020]

Learning dot product using MLP is non-trivial



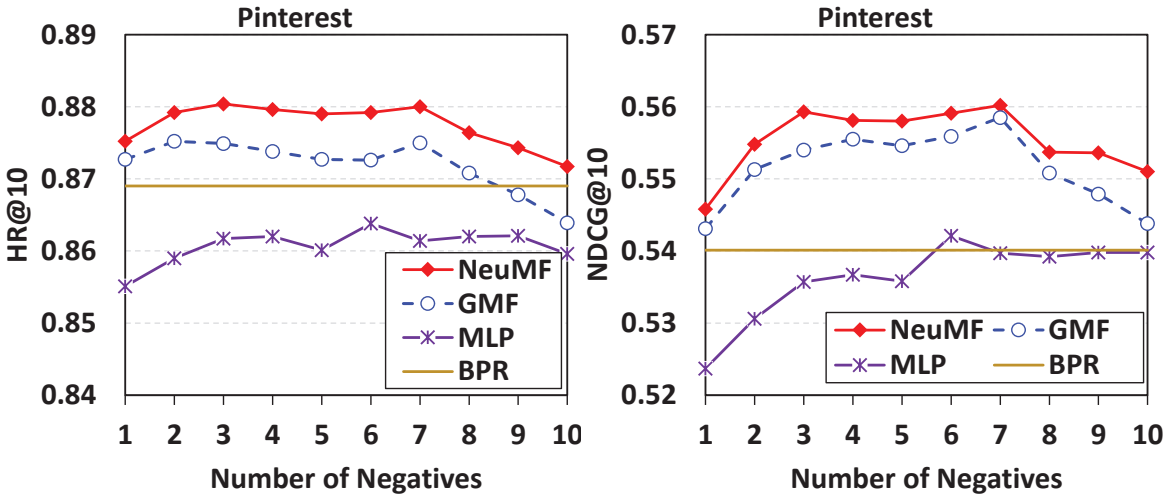
[Rendle, S., Krichene, W., Zhang, L., & Anderson, J., 2020]

Impact of Enhancements

Impact of 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

Impact of Negative Sampling



(c) Pinterest — HR@10 (d) Pinterest — NDCG@10

[He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S., 2017]

Note on Reproducibility

- **Neural collaborative filtering is very active research area**
  - Many complex approaches (compared to the simple MF)

- **A study tried to reproduce the reported results for 26 papers**
  - Only 12/16 papers had reproducible setup (code + dataset)
  - “None of the computationally complex neural methods was actually consistently better than already existing learning-based techniques, e.g., using matrix factorization or linear models”

- **You can test this in the assignment!**

[Ferrari Dacrema, M., Boglio, S., Cremonesi, P., & Jannach, D., 2021]

Summary

- **Neural architectures for collaborative filtering:**
  - AutoRec: autoencoder-based
  - NCF: MLP + (G)MF

- **Deeper more expressive models can improve performance**
  - Combining with MF can help generalization

- **In Assignment 5:**
  - Implement and evaluate NCF on MovieLens (explicit ratings)
  - Compare with Funk's SVD