Neural Graph Collaborative Filtering

Youngbin Kim

Seoul National University

SIGIR 2019

Collaborative Filtering

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	✓			
Panda Bear	✓	✓		✓
Grizzly Bear			✓	
Chloe Park		✓		✓

Collaborative Filtering

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	✓			
Panda Bear	✓	✓		✓
Grizzly Bear			✓	
Chloe Park	**Recommend!**	✓		✓

Collaborative Filtering

- Matrix Factorization

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	✓			
Panda Bear	✓	✓		✓
Grizzly Bear			✓	
Chloe Park		✓		✓

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	1	0	0	0
Panda Bear	1	1	0	1
Grizzly Bear	0	0	1	0
Chloe Park	0	1	0	1

User-Item interaction Matrix

Collaborative Filtering

- Matrix Factorization

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	1	0	0	0
Panda Bear	1	1	0	1
Grizzly Bear	0	0	1	0
Chloe Park	0	1	0	1

User-Item interaction Matrix

≈

User embedding

0.3256	0.5755
-0.7000	0.6974
1.048	-0.3193
-0.7106	0.6467

Item embedding

0.2600	-1.445	0.9332	0.2381
1.564	-0.0526	-0.0707	1.765

	Little Women	Jojo Rabbit	1917	The Man Standing Next
Ice Bear	0.9851	-0.5009	0.2632	1.093
Panda Bear	0.9091	0.9749	-0.7024	1.064
Grizzly Bear	-0.2271	-1.498	1.000	-0.3141
Chloe Park	0.8270	0.9929	-0.7088	0.9723

Problem Definition

- Given
  - Set of users $\mathcal{U} \in \mathbb{R}^N$
  - Set of items $\mathcal{I} \in \mathbb{R}^M$
  - User-item interaction matrix $\boldsymbol{R} \in \mathbb{R}^{N \times M}$,
    - Each entry $r_{ui} = 0$ (otherwise) or 1 (there is interaction)
- Find
  - Recommend top-K preference items for a user

Motivations

- Limitation of existing methods:
  - Do not fully explore the high-order connectivity
  - Build the embedding function with the descriptive features(e.g., IDs or attributes) only

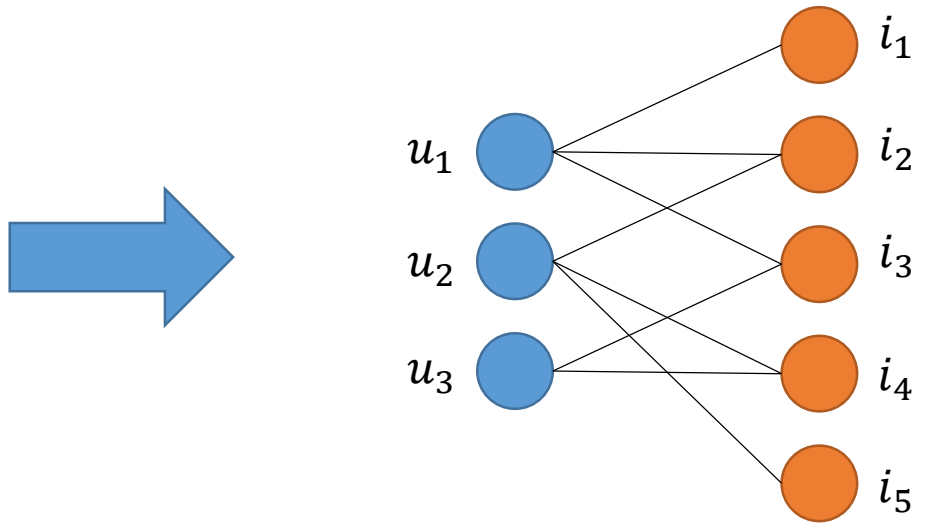
- NGCF(Neural Graph Collaborative Filtering)
  - Can capture high-order connectivity information by using graph neural network
  - Explicitly exploit the collaborative signal in the embedding function

Motivations

- NGCF
  - Can capture high-order connectivity information by using graph neural network

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

User-Item interaction Matrix $\boldsymbol{R}$



User-Item interaction Graph

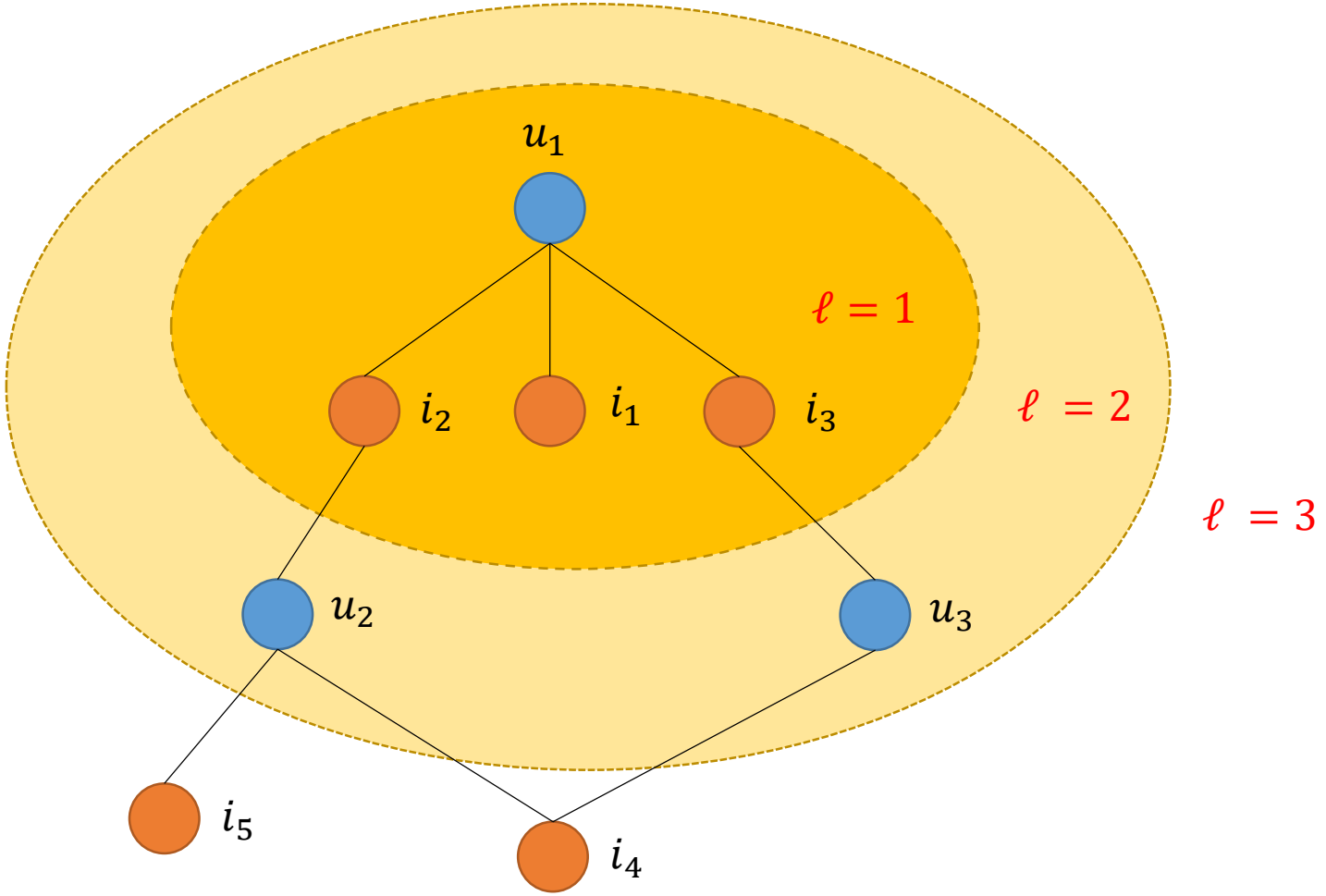
Adjacency matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & \boldsymbol{R} \\ \boldsymbol{R}^T & 0 \end{bmatrix}$$

All zero matrix

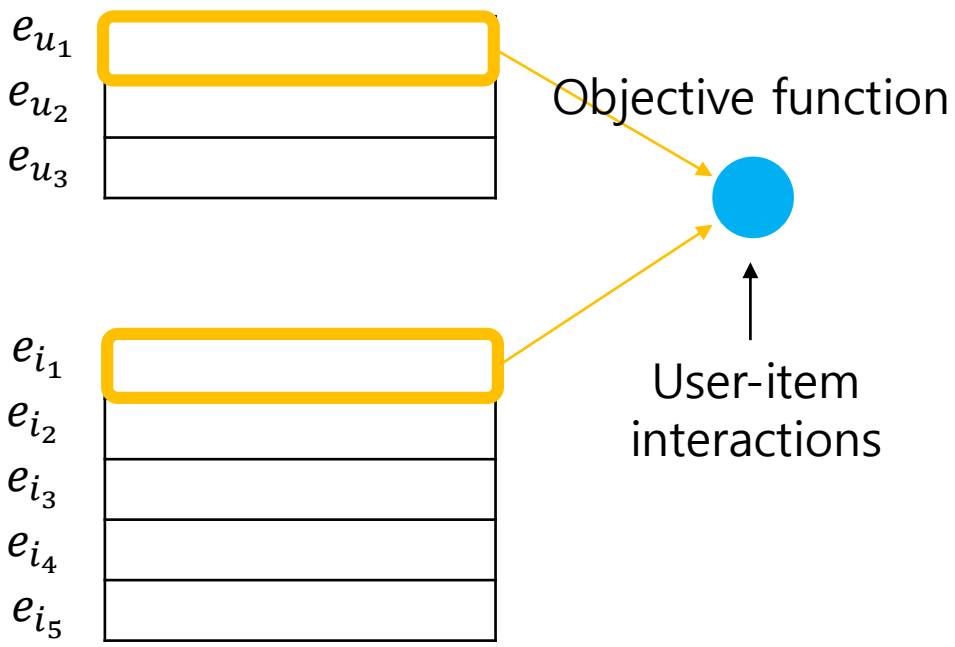
Motivations

- NGCF
  - Can capture high-order connectivity information by using graph neural network

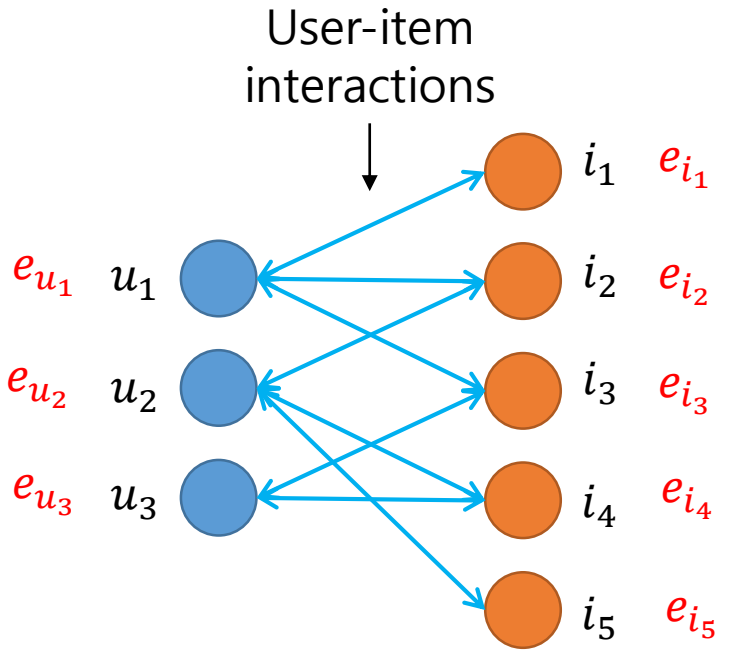


Motivations

- NGCF
  - Explicitly exploit the collaborative signal in the embedding function
    - Collaborative signal: latent in user-item interactions to reveal the behavioral similarity between users (or items)



VS.

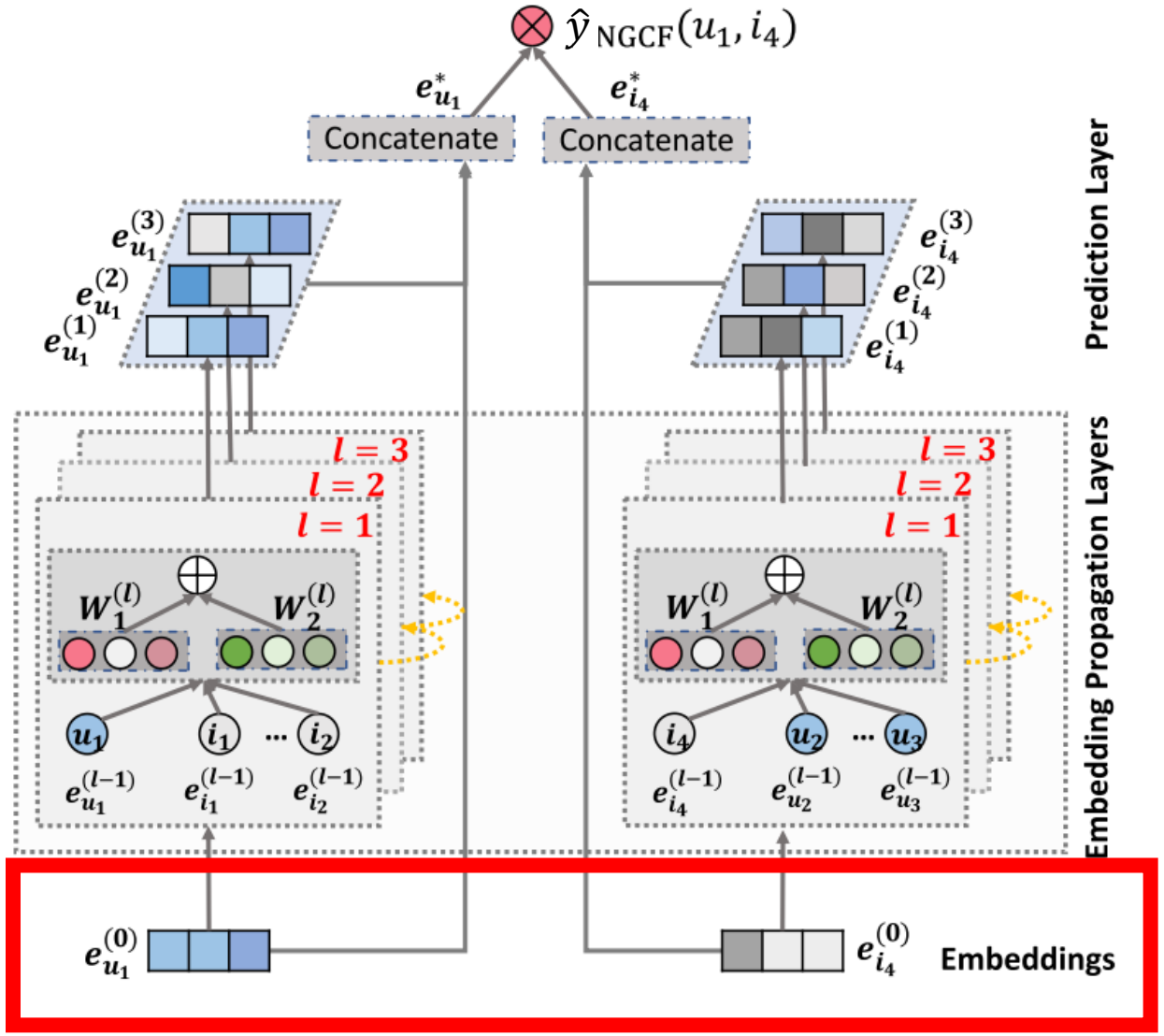


Existing methods

NGCF

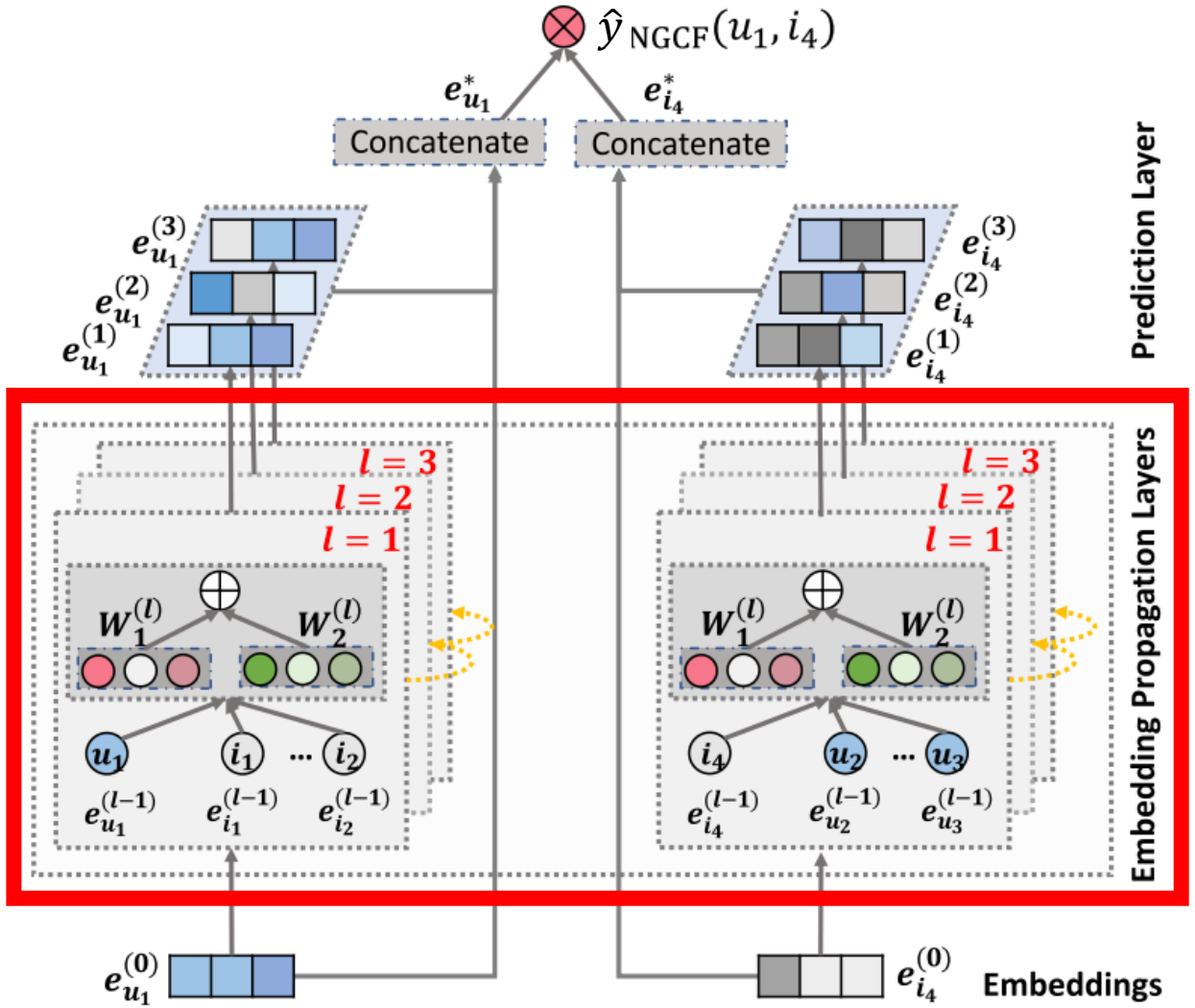
Embedding Layer

- Embedding look-up table
  - $\boldsymbol{E} = [\boldsymbol{U}, \boldsymbol{V}]^T \in \mathbb{R}^{(N+M)\times d}$
    - $\boldsymbol{U} \in \mathbb{R}^{N\times d}$: user embedding matrix
      - $e_u \in \boldsymbol{U}$: user embedding vector
      - d : embedding size
    - $\boldsymbol{V} \in \mathbb{R}^{M\times d}$: item embedding matrix
      - $e_i \in \boldsymbol{V}$: item embedding vector



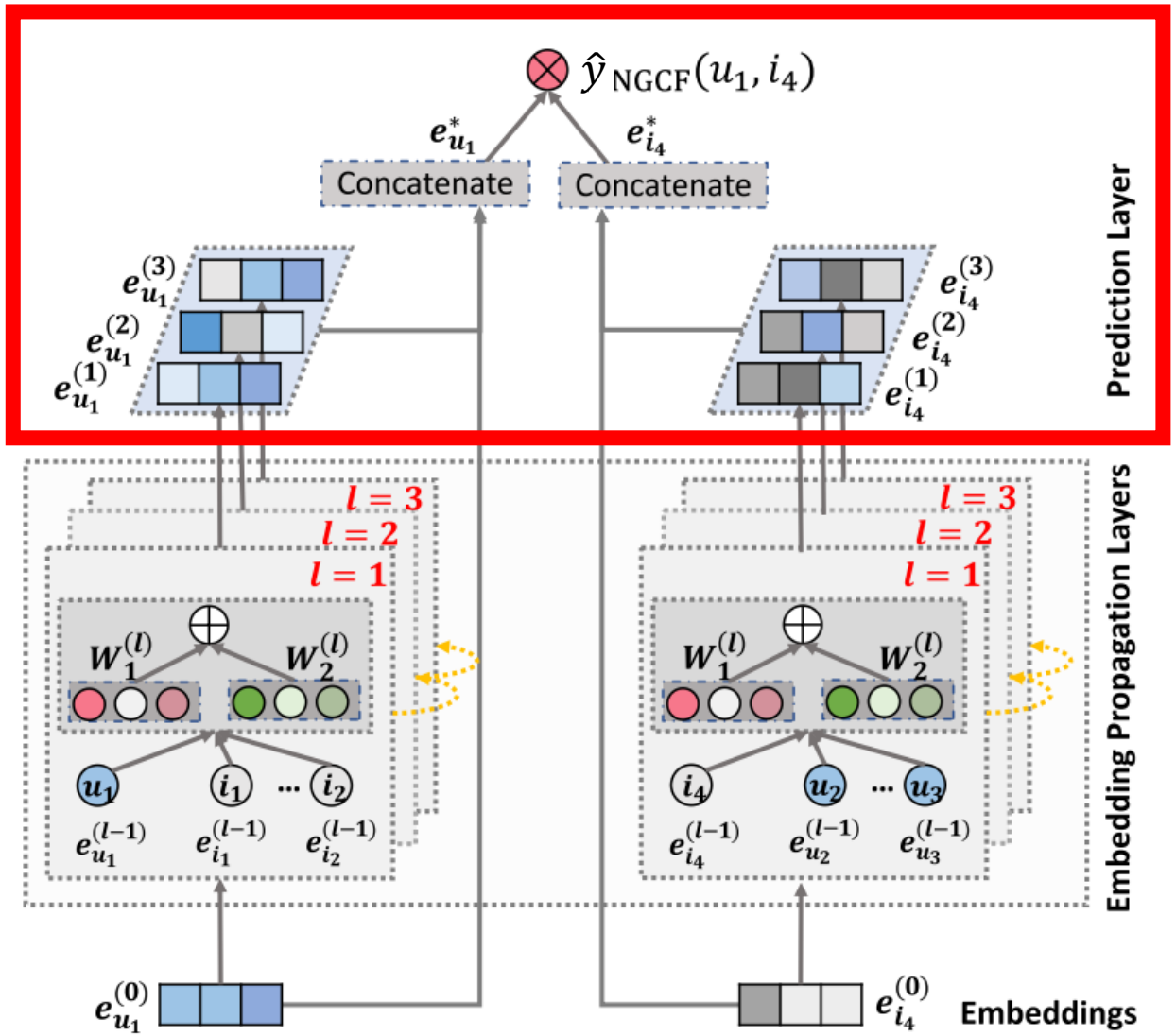
Embedding Propagation Layers

- Matrix Form
  - $\boldsymbol{E}^{(l)} = \sigma((\boldsymbol{\mathcal{L}} + \boldsymbol{I})\boldsymbol{E}^{(l-1)}\boldsymbol{W}_1^{(l-1)} + \boldsymbol{\mathcal{L}}\boldsymbol{E}^{(l-1)} \odot \boldsymbol{E}^{(l-1)}\boldsymbol{W}_2^{(l-1)})$
    - $\boldsymbol{\mathcal{L}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$
      - $\boldsymbol{D}$: Diagonal matrix
    - $\sigma(\cdot)$: activation function
      - LeakyReLU
    - $\odot$: element-wise product
    - $\boldsymbol{I}$: identity matrix



Prediction Layer

- Final embedding Look-up table
  - $\boldsymbol{E}^* = \boldsymbol{E}^{(0)}||\cdots||\boldsymbol{E}^{(L)} = [\boldsymbol{U}^*, \boldsymbol{V}^*]^T$
- Model prediction
  - $\hat{y}_{NGCF}(u,i) = {e_u^*}^T e_i^*$
    - $e_u^* \in \boldsymbol{U}^*$: Final user embedding vector
    - $e_i^* \in \boldsymbol{V}^*$: Final item embedding vector
- Loss function
  - Pairwise BPR loss
    - $Loss = \sum_{(u,i,j\in\mathcal{O})} -ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\|\Theta\|_2^2$
      - $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$



Experiment Results

Table 1: Statistics of the datasets.

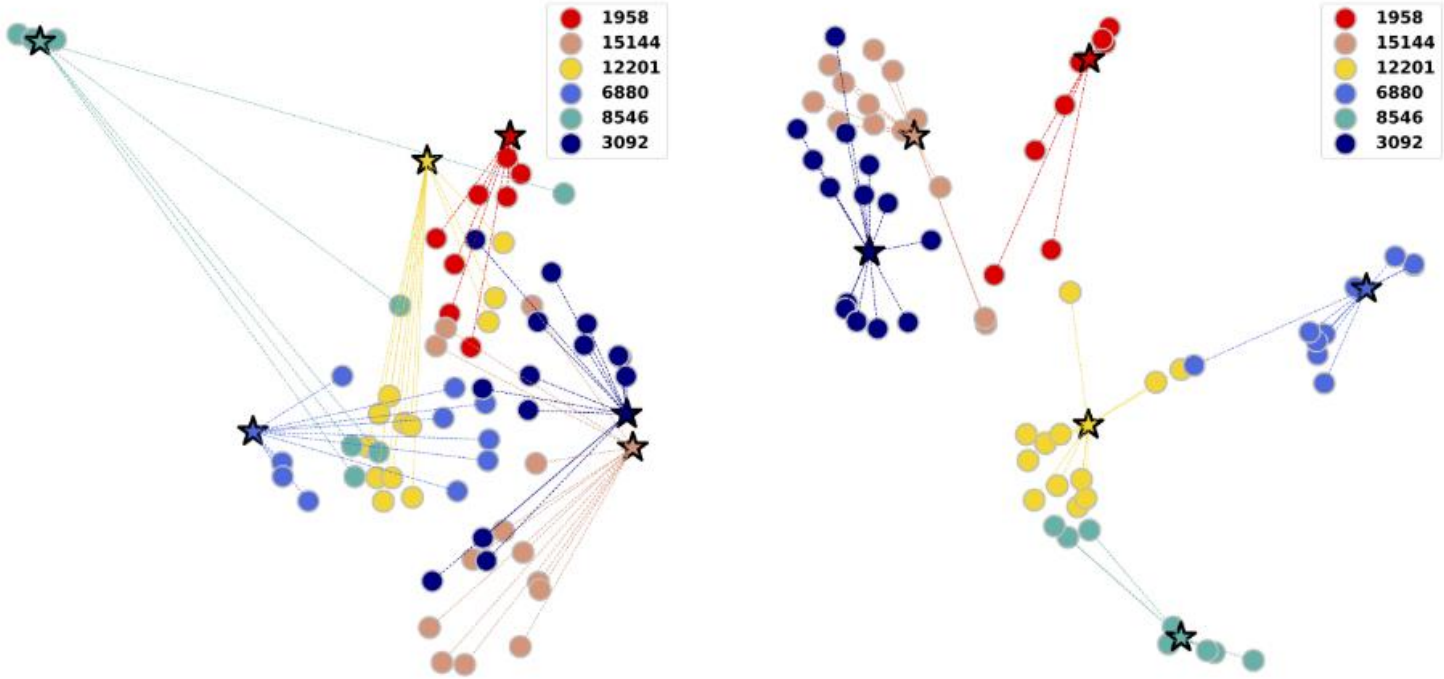
Dataset	#Users	#Items	#Interactions	Density
Gowalla	29, 858	40, 981	1, 027, 370	0.00084
Yelp2018	31, 831	40, 841	1, 666, 869	0.00128
Amazon-Book	52, 643	91, 599	2, 984, 108	0.00062

Table 2: Overall Performance Comparison.

	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
MF	0.1291	0.1878	0.0317	0.0617	0.0250	0.0518
NeuMF	0.1326	0.1985	0.0331	0.0840	0.0253	0.0535
CMN	**0.1404**	**0.2129**	0.0364	0.0745	0.0267	0.0516
HOP-Rec	0.1399	0.2128	**0.0388**	**0.0857**	**0.0309**	**0.0606**
GC-MC	0.1395	0.1960	0.0365	0.0812	0.0288	0.0551
PinSage	0.1380	0.1947	0.0372	0.0803	0.0283	0.0545
NGCF	**0.1547***	**0.2237***	**0.0438***	**0.0926***	**0.0344***	**0.0630***
%Improv.	10.18%	5.07%	12.88%	8.05%	11.32%	3.96%
p-value	1.01e-4	5.38e-3	4.05e-3	2.00e-4	4.34e-2	7.26e-3

Experiment Results

- Effect of high-order connectivity
  - The connectivity of users and items is well reflected in the embedding space



(a) MF (NGCF-0)

(b) NGCF-3

☆ : users from Gowalla
○ : items from Gowalla

Conclusion

- NGCF explicitly incorporated collaborative signal into the embedding function
- NGCF leverages high-order connectivity in the user-item interaction graph
- Experiments on the three real-world datasets demonstrate effectiveness of the user-item interaction graph structure into the embedding learning process


Thank
you

Reproduced Results

- 저자 코드는 tensorflow로 구현되어 있어 이를 pytorch로 reproduce 함
- 재현 코드 링크: https://github.com/immabe/NGCF_pytorch
- 실행 방법
  - Environments setting(using conda, linux)
    - Gpu가 있고, cuda와 cudnn이 설치되었다고 가정

```
conda create -n ngcf
conda activate ngcf
conda install pytorch torchvision cudatoolkit=10.2 -c pytorch
conda install -c conda-forge tqdm
conda install -c anaconda scikit-learn
conda install -c anaconda scipy
```

- 실행방법(cont.)
  - Download code

    `git clone https://github.com/immabe/NGCF_pytorch.git`
  - Train
    - NGCF 폴더로 이동

      `python NGCF_pytorch.py`
    - Train option 설정
      - NGCF/utility/ 폴더의 parser_pytorch.py에서 설정
      - 실행 시 설정

```
python NGCF_pytorch.py --dataset gowalla --regs [1e-5] --embed_size 64 --layer_size [64,64,64] --lr 0.0001 --save_flag 1 --pretrain 0 --batch_size 1024 --epoch 400 --verbose 1 --node_dropout [0.1] --mess_dropout [0.1,0.1,0.1]
```

  - Test
    - 미리 train한 모델이 NGCF_pytorch/NGCF/model 폴더에 있음

```
python NGCF_pytorch.py --dataset gowalla --regs [1e-5] --embed_size 64 --layer_size [64,64,64] --lr 0.0001 --save_flag 0 --pretrain 1 --batch_size 1024 --epoch 400 --verbose 1 --node_dropout [0.1] --mess_dropout [0.1,0.1,0.1] --pretrain_path 'model/model_gowalla.pth'
```

Reproduced Results

- 실험결과

NGCF_pytorch 실험 결과 console 출력

Dataset	NGCF		NGCF_pytorch	
	recall	ndcg	recall	ndcg
Gowalla	0.1547	0.2237	0.1461	0.2780

- NGCF: 논문에 제시된 실험 결과
- NGCF_pytorch: pytorch로 재현한 NGCF
- NGCF_pytorch가 논문에 제시된 결과보다 recall은 약 5%정도 낮게 재현되었고, ndcg는 약 24%정도 높게 재현되었다. 실험 결과의 차이는 아래와 같은 이유들로 보인다.
  - Tensorflow와 pytorch의 구현상의 차이
  - NGCF_pytorch 실험을 한번만 수행
  - Sample된 train data나 test data의 차이