Machine Learning with Graphs (MLG)

RecSys with Graph Neural Networks

Utilize GNN to learn high-order interactions

Part 1

Cheng-Te Li (李政德)

Institute of Data Science
National Cheng Kung University
chengte@mail.ncku.edu.tw

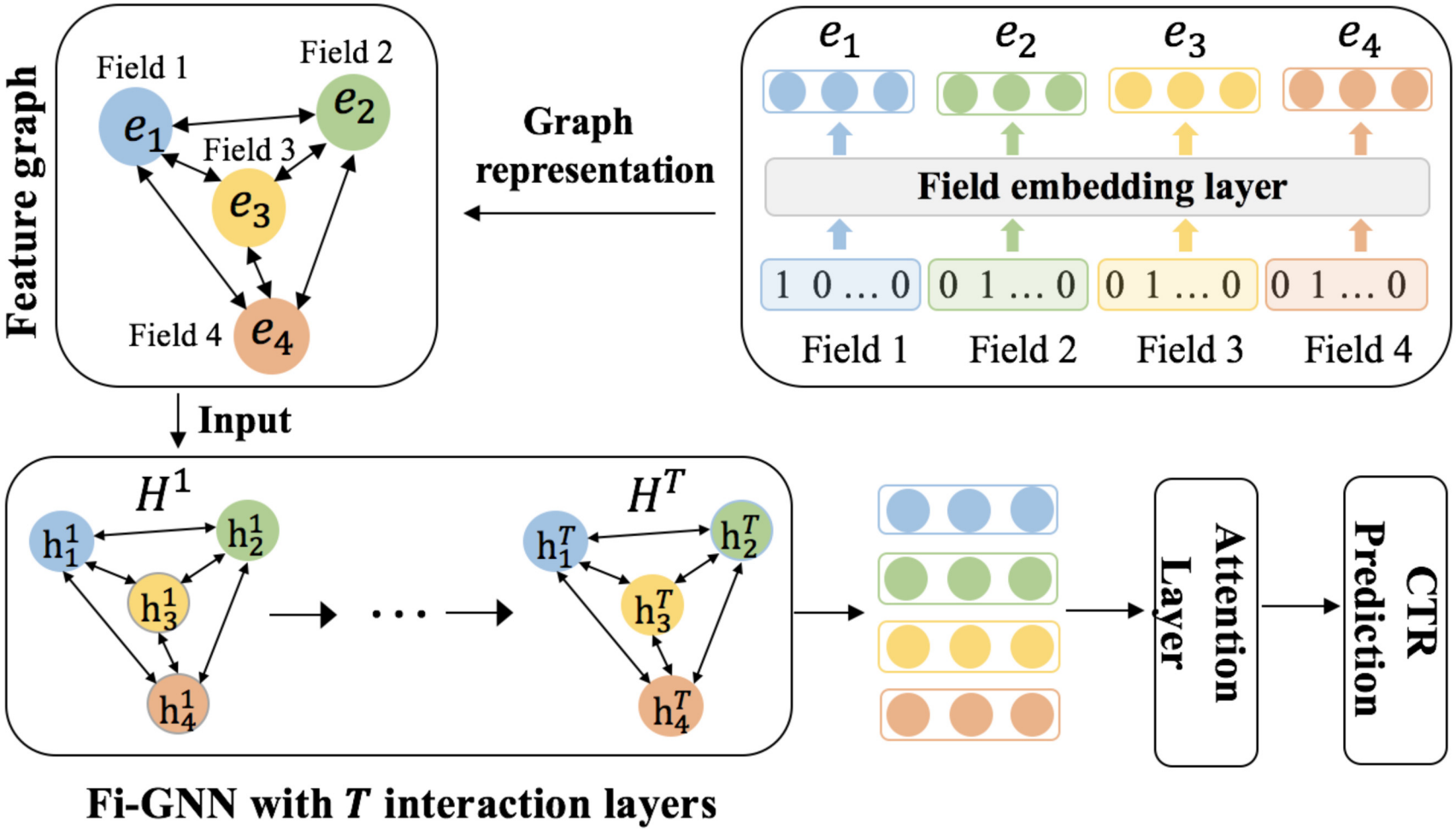
Feature Interaction GNN

Given a user and the page she is visiting, the goal is to predict the probability that she will click on a given ad

Multi-field categorical data. For example,
the four-fields categorical features for movies:
(1) Language = {English, Chinese, Japanese, ... }
(2) Genre = {action, fiction, ... }
(3) Director = {Ang Lee, Christopher Nolan, ... }
(4) Starring = {Bruce Lee, Leonardo DiCaprio, ... }

Conventional approach: converted to sparse one-hot encoding vectors, and then embedded to dense real-value vectors, which can be used to model feature interactions via concatenation, dot product, and hidden layer

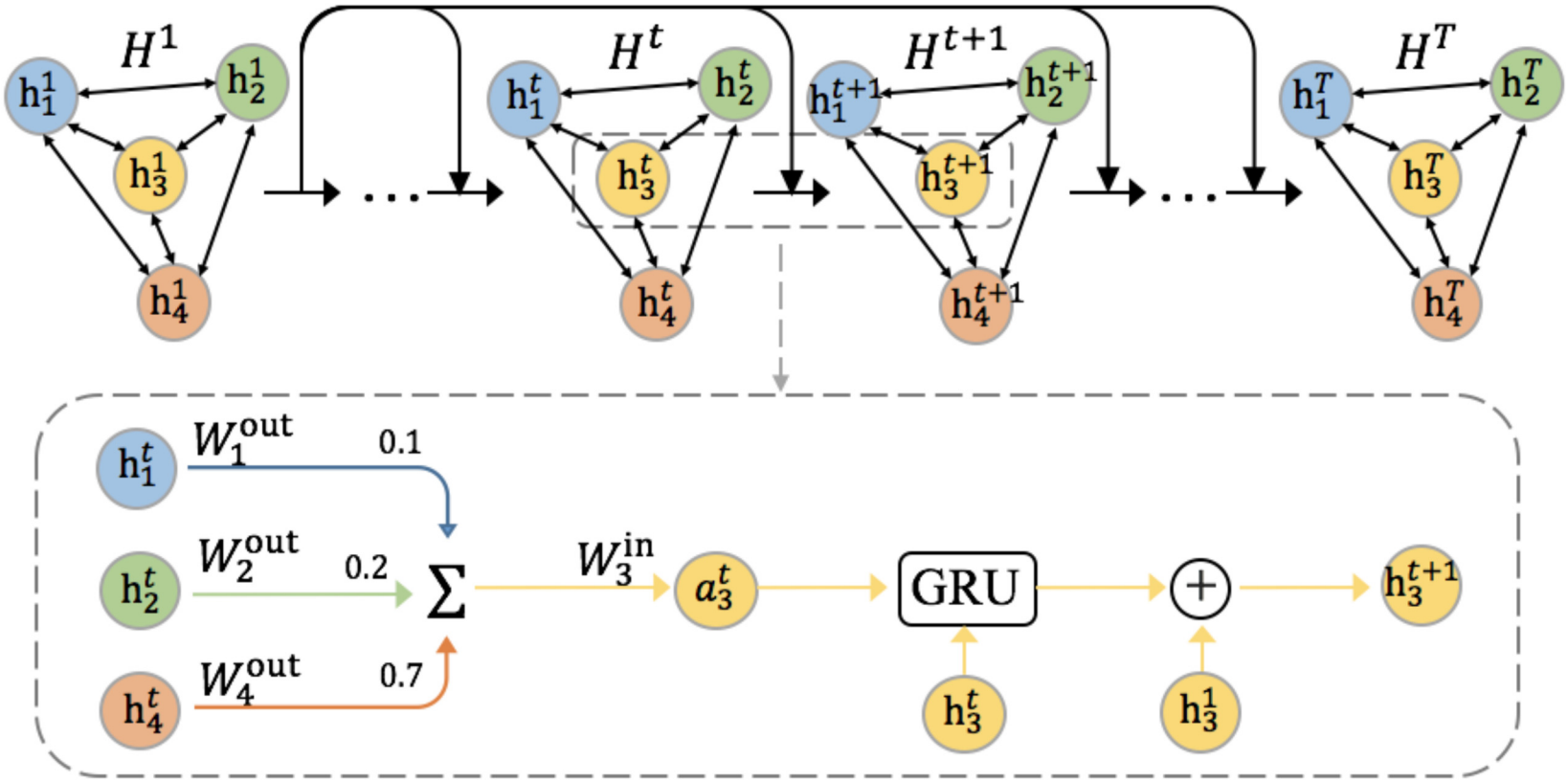
Feature Interaction GNN



(1) The input multi-field feature vector is converted to field embedding vectors via an embedding layer and represented as a feature graph

(2) Feature graph is fed into **Fi-GNN** to model feature interactions

(3) An attention layer is applied on Fi-GNN output to predict CTR



$$\mathbf{a}_i^t = \sum_{n_j \to n_i \in \mathcal{E}} \mathbf{A}[n_j, n_i]\mathbf{W}_{out}^j \mathbf{W}_{in}^i \mathbf{h}_j^{t-1} + \mathbf{b}_p$$

Graph attention

$$\mathbf{h}_i^t = GRU(\mathbf{h}_i^{t-1}, \mathbf{a}_i^t) + \mathbf{h}_i^1$$

$$\hat{y}_i = MLP_1(\mathbf{h}_i^p)) \qquad \hat{y} = \sum_{i=1}^{m} a_i \hat{y}_i$$

$$a_i = MLP_2(\mathbf{h}_i^p))$$

State vector of node n_i is updated via GRU based on the aggregated state information at i and its state at last step

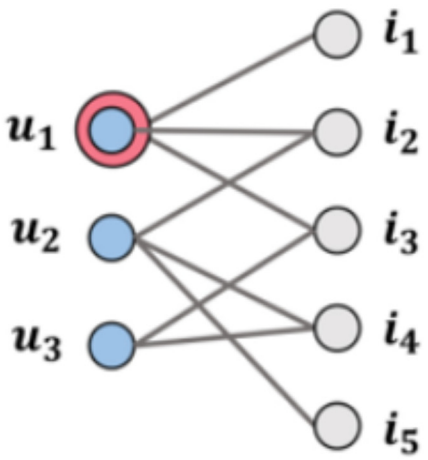
Add extra residual connections to update note states (combine the low-order and high-order interactions)

Neural Graph Collaborative Filtering (NGCF)

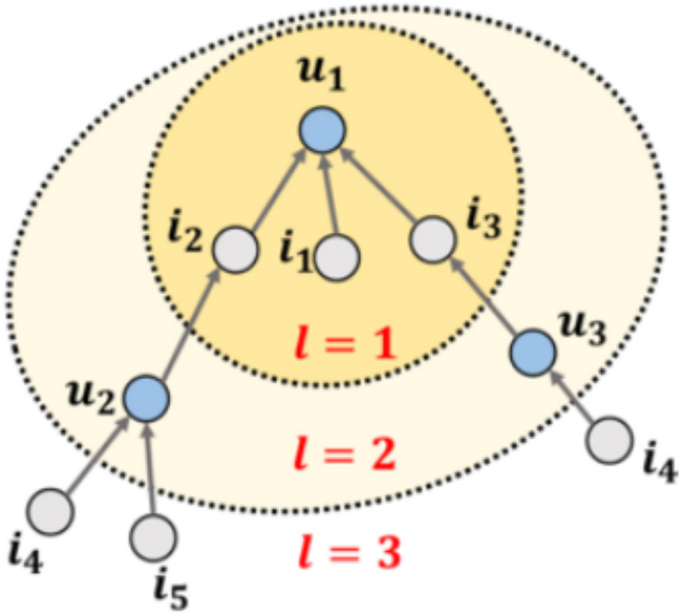
- Revisit CF via high-order connectivity
  - The paths that reach u_1 from any node with the path length l larger than 1
  - A natural way to encode collaborative signal in the interaction graph structure

Why u_1 may like i_4

- $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$
- $u_1 \leftarrow i_3 \leftarrow u_3 \leftarrow i_4$



User-Item Interaction Graph

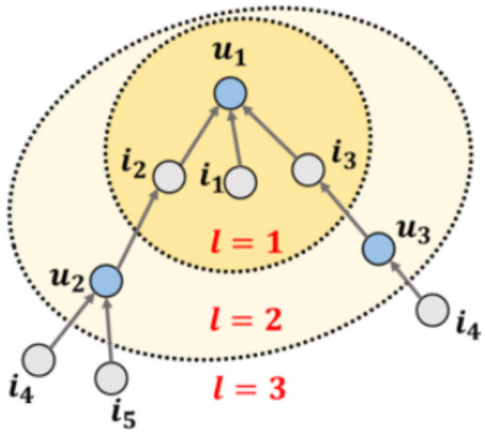


High-order Connectivity for u_1

NGCF

- Inspired by GNNs
  - 1. Propagate embeddings recursively on the user-item graph
  - 2. Construct information flows in the embedding space

- Comp.1: Information Construction:



message passed from i to u

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\right)$$

discount factor

- message dependent on the affinity, distinct from GCN, GraphSage, etc.
- Pass more information to similar nodes

- Comp.2 & 3: Neighbor Aggregation & Representation Update:

$$\mathbf{e}_u^{(1)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u} + \sum_{i\in\mathcal{N}_u}\mathbf{m}_{u\leftarrow i}\right)$$

self-connections

all neighbors of u

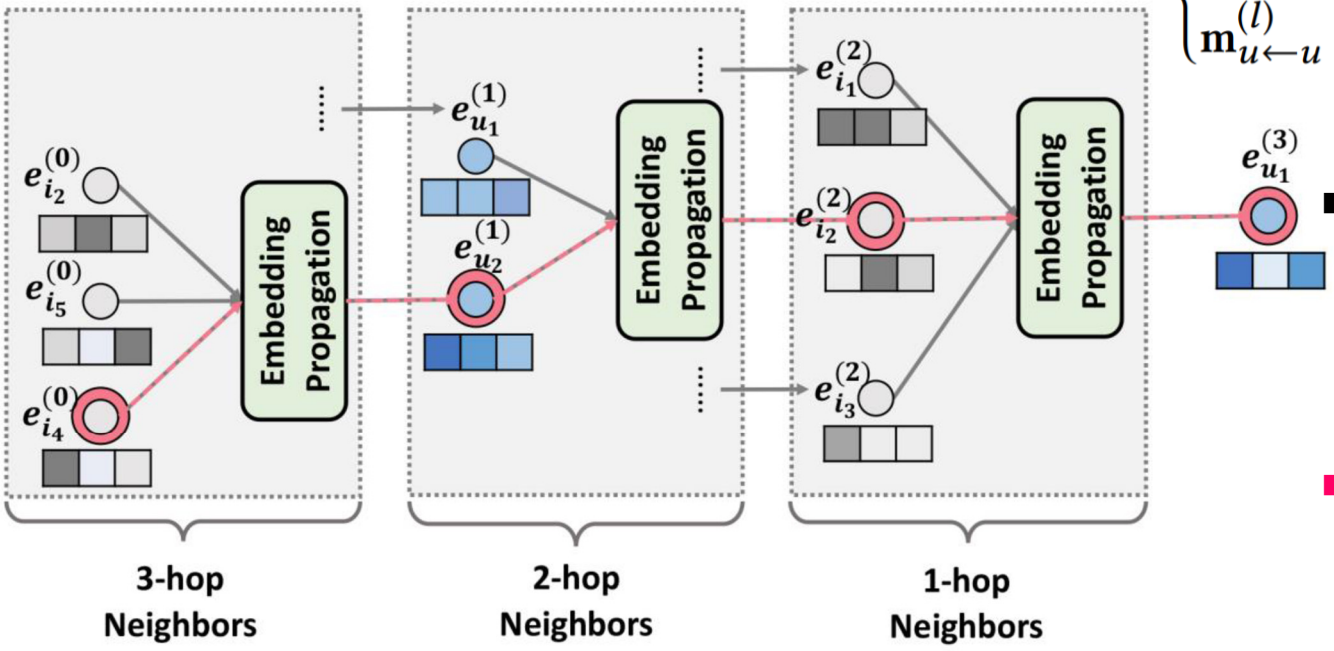
High-order Propagation in NGCF

- Stack more embedding propagation layers to explore the high-order connectivity info

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\right)$$

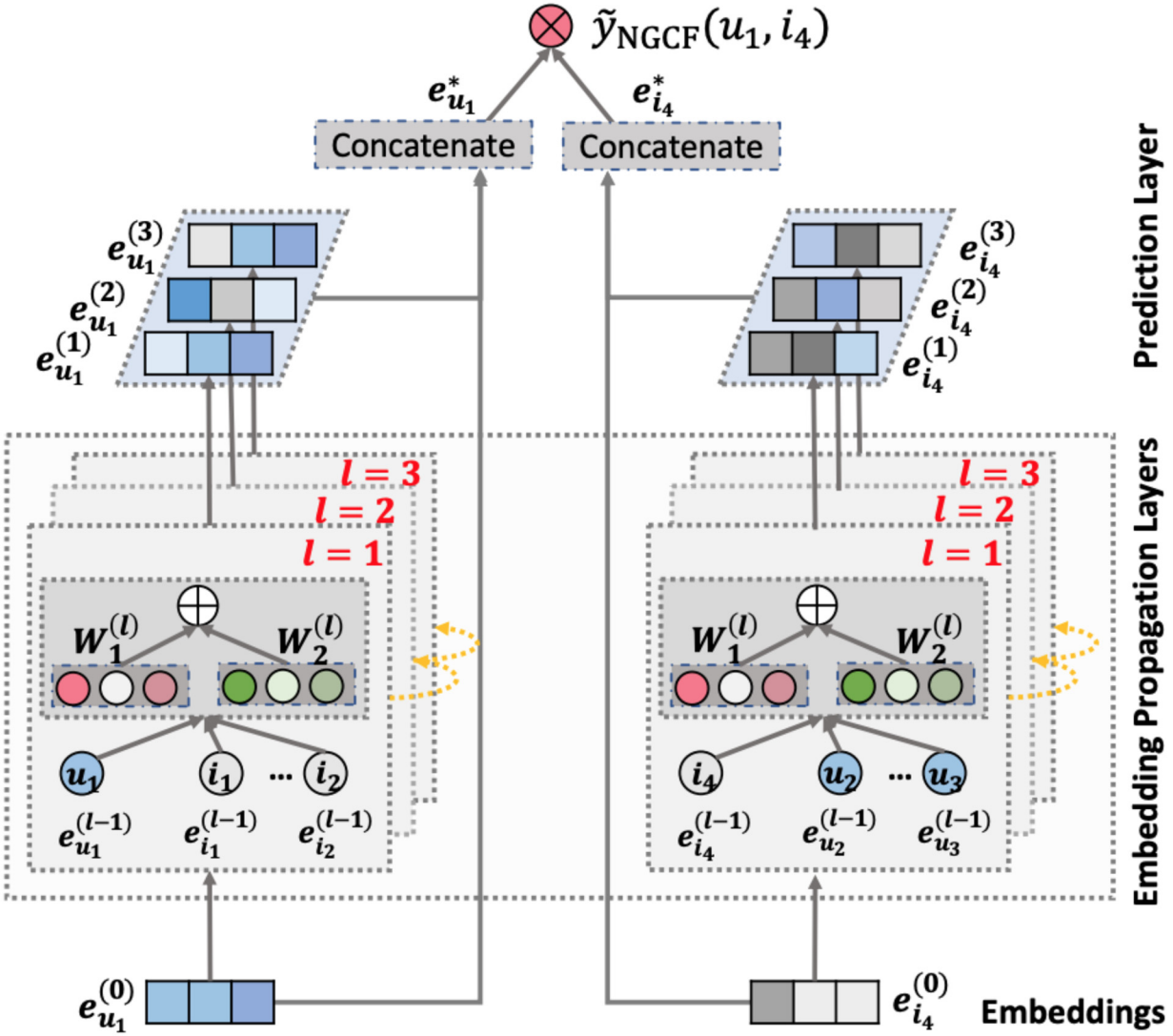
representation of u at the l-th layer

$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right) \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases} \qquad p_{ui} = \frac{1}{\sqrt{|N_u||N_i|}}$$



- Collaborative signal like $u1 \leftarrow i2 \leftarrow u2 \leftarrow i4$ can be captured in embedding propagation process
- Collaborative signal can be injected into representation learning process

Overall NGCF Framework



$$\mathbf{e}^*_u = \mathbf{e}^{(0)}_u \| \cdots \| \mathbf{e}^{(L)}_u$$

$$\mathbf{e}^*_i = \mathbf{e}^{(0)}_i \| \cdots \| \mathbf{e}^{(L)}_i$$

$$\hat{y}_{\text{NGCF}}(u, i) = \mathbf{e}^{*\top}_u \mathbf{e}^*_i$$

The representations at different layers

- emphasize the messages passed over different connections
- have different contributions in reflecting user preference

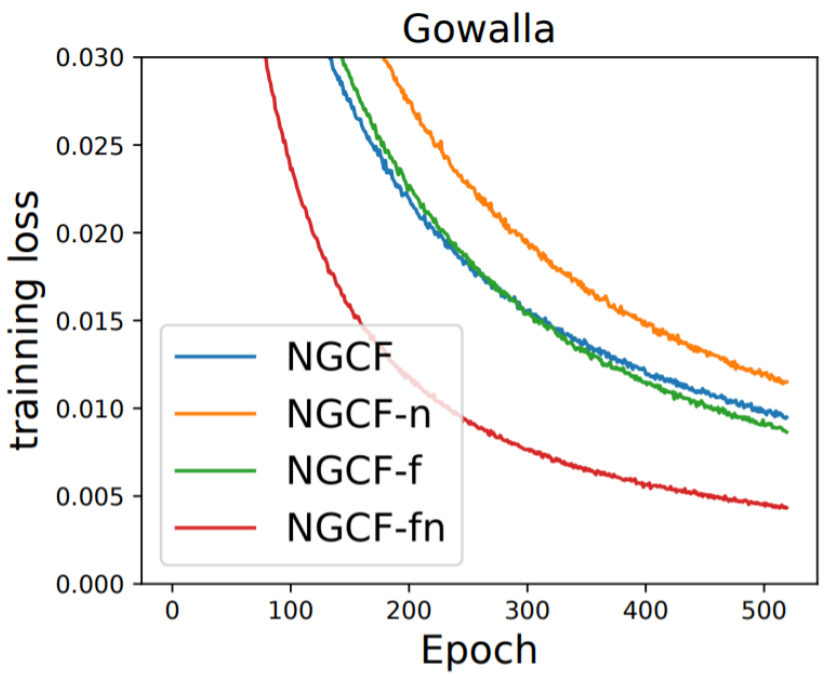
Argument on NGCF

- Designs of NGCF are rather heavy and burdensome
  - Many operations are directly inherited from GCN w/o justification

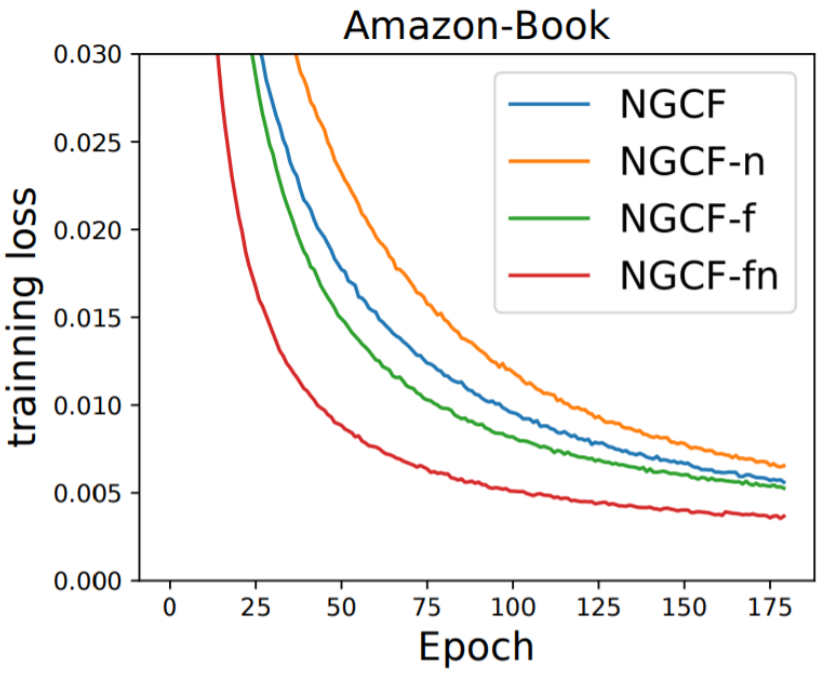
	GNNs	NGCF
Original task	Node classification	Collaborative filtering
Input data	Rich node features • Attributes, text, image data	Only node ID • One-hot encoding
Feature transformation	Distill useful information	Generate ID embeddings
Neighborhood aggregation	Pass messages from neighbors to the egos	Pass messages from neighbors to the egos
Nonlinear activation	Enhance representation ability	Negatively increases the difficulty for model training

Empirical Evidence on Training Difficulty

- Removing feature transformation (NGCF-f) → decrease training loss
- Removing nonlinear activation (NGCF-n) → increase training loss
- But, removing nonlinear activation & feature transformation (NGCF-fn) → significantly decrease training loss



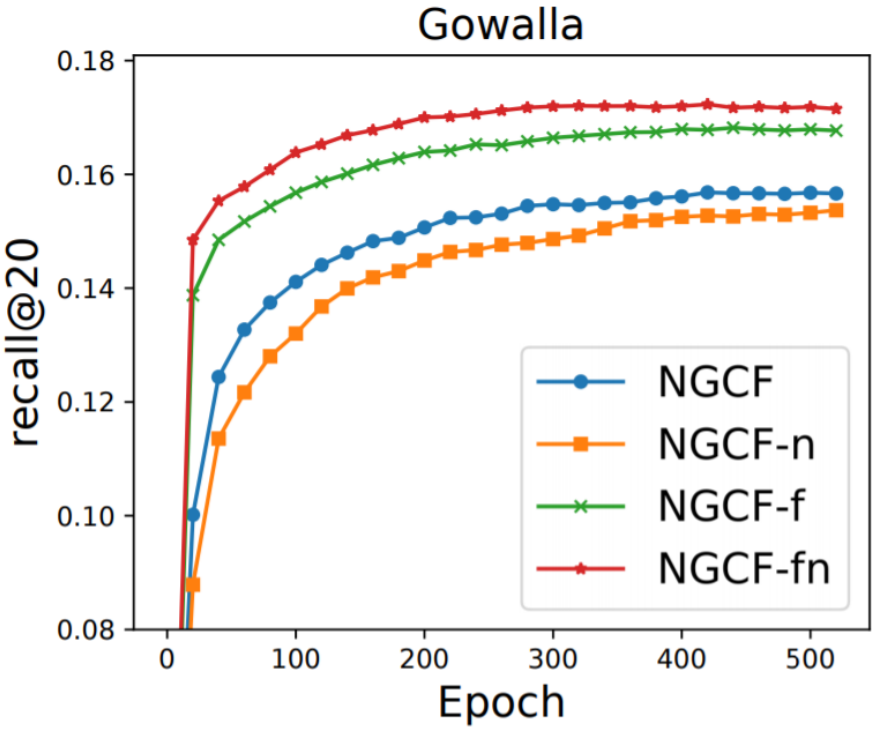
(a) Training loss on Gowalla



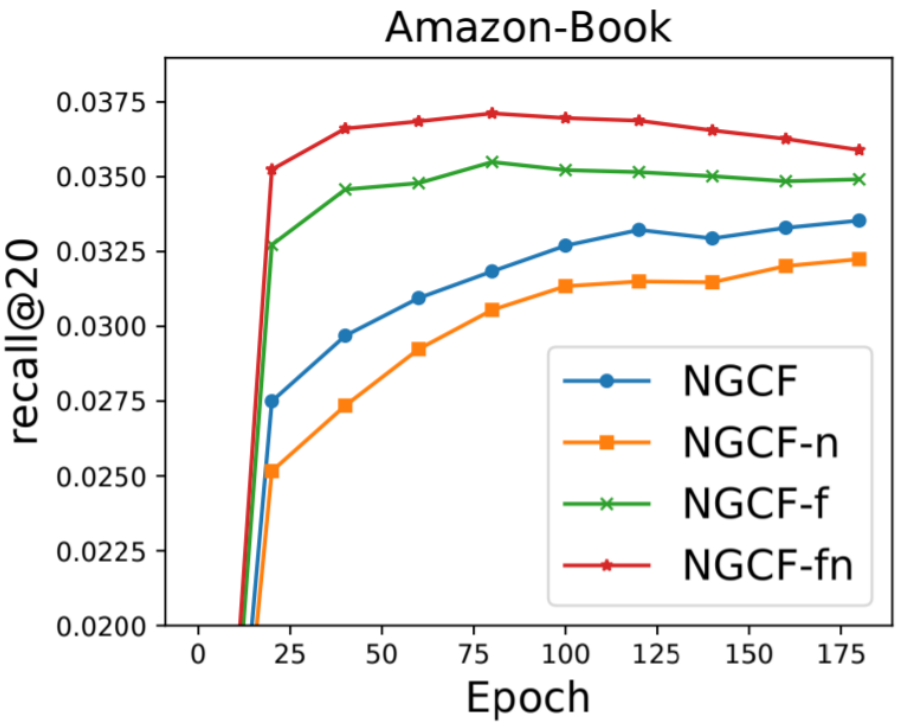
(c) Training loss on Amazon-Book

Empirical Evidence on Training Difficulty

- Removing feature transformation (NGCF-f) → improve testing acc
- Removing nonlinear activation (NGCF-n) → hurt testing accuracy
- But, removing nonlinear activation & feature transformation (NGCF-fn) → significantly improve testing accuracy



(b) Testing recall on Gowalla



(d) Testing recall on Amazon-Book

Light Graph Convolution

NGCF

- Graph Convolution Layer

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\right)$$

- Layer Combination

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \| \cdots \| \mathbf{e}_u^{(L)}$$

- Matrix Form

$$\mathbf{E}^{(l)} = \text{LeakyReLU}\left((\mathcal{L} + \mathbf{I})\mathbf{E}^{(l-1)}\mathbf{W}_1^{(l)} + \mathcal{L}\mathbf{E}^{(l-1)} \odot \mathbf{E}^{(l-1)}\mathbf{W}_2^{(l)}\right)$$

LightGCN

- **Light** Graph Convolution Layer

$$\mathbf{e}_u^{(k+1)} = \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}$$

- Layer Combination

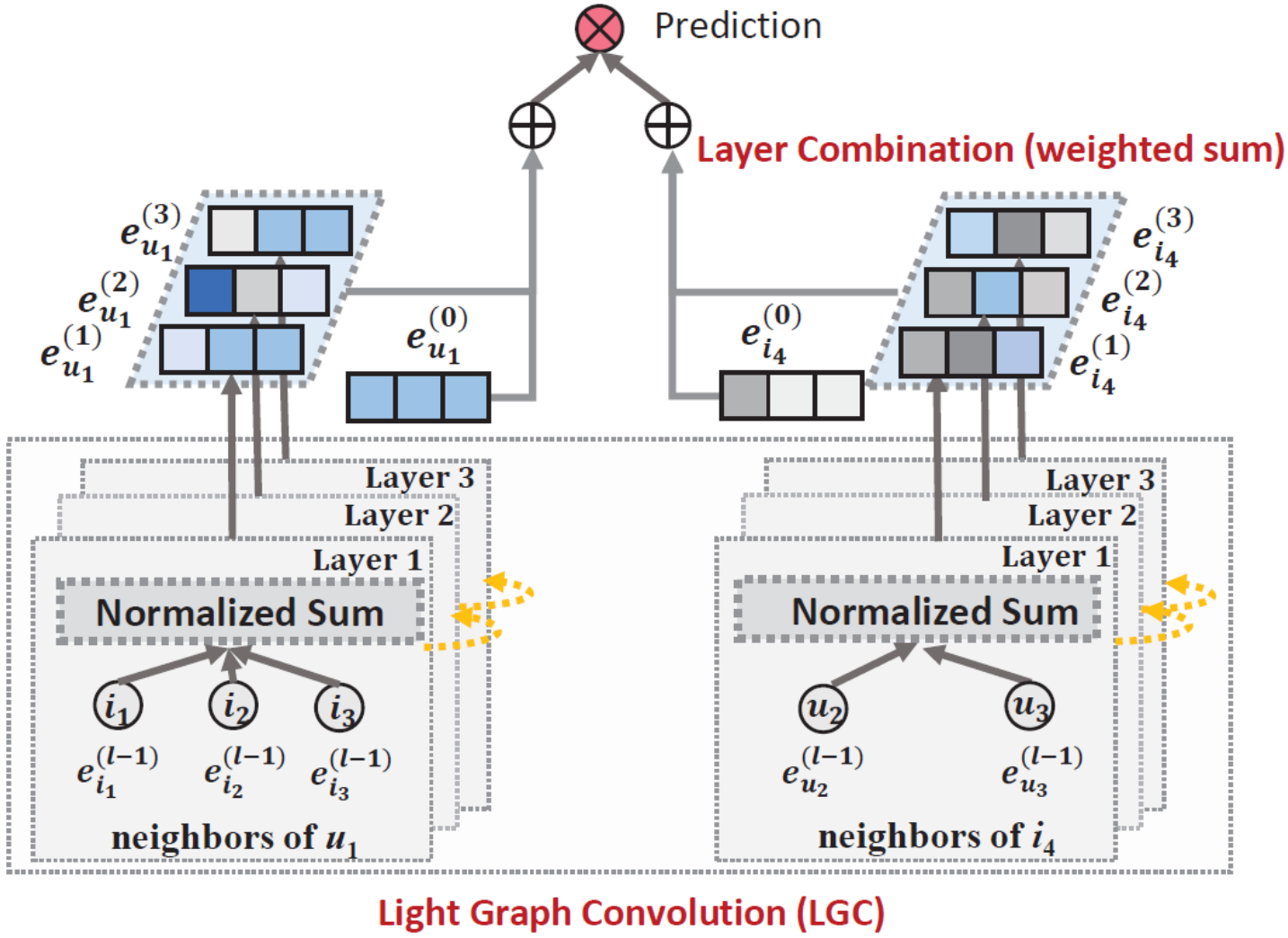
$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}$$

- Matrix Form

$$\mathbf{E}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)}$$

- Only simple weighted sum aggregator is remained
  - No feature transformation
  - No nonlinear activation
  - No self connection

LightGCN Framework



$$\mathbf{E} = \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)}$$
$$= \alpha_0\mathbf{E}^{(0)} + \alpha_1\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \alpha_K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}$$

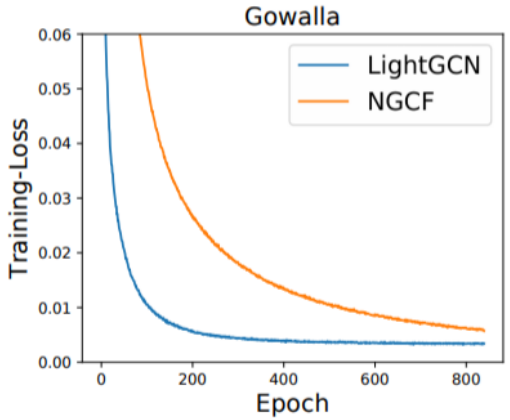
$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

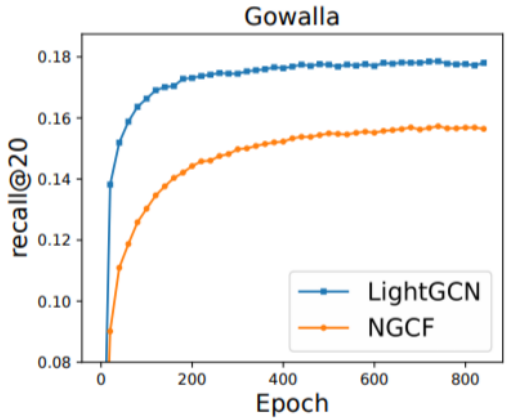
α: importance of the k-th layer embedding in constituting the final embedding. “avg” or “attention”

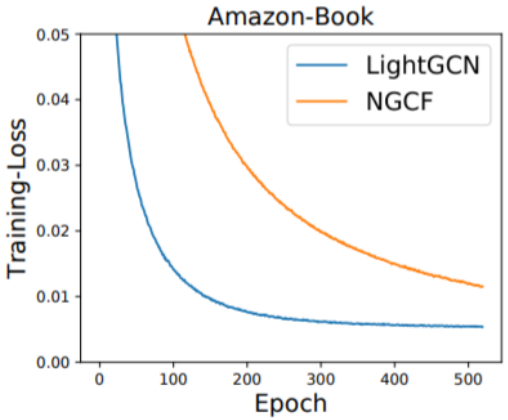
Performance: LightGCN vs. NGCF

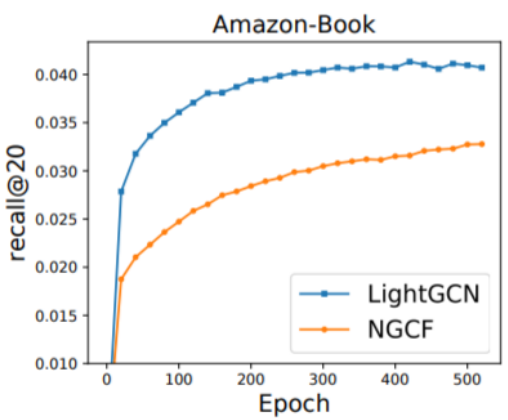
Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)









References

- Z. Li et al. “**Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction**” ACM CIKM 2019 **51 cites**
- X. Wang et al. “**Neural Graph Collaborative Filtering**” ACM SIGIR 2019 **525 cites**
- X. He et al. “**LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation**” ACM SIGIR 2020 **263 cites**