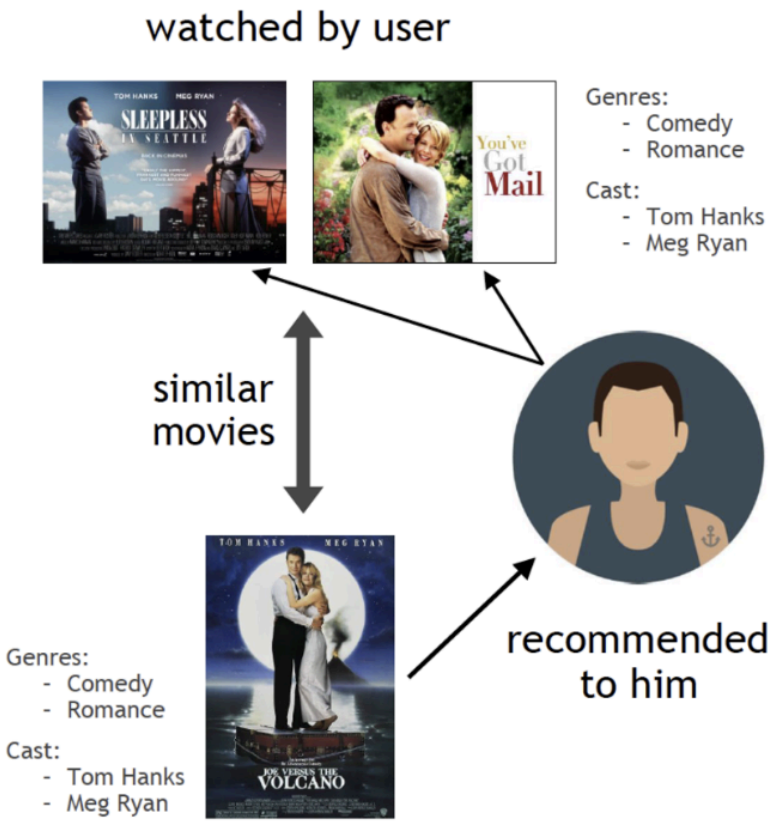
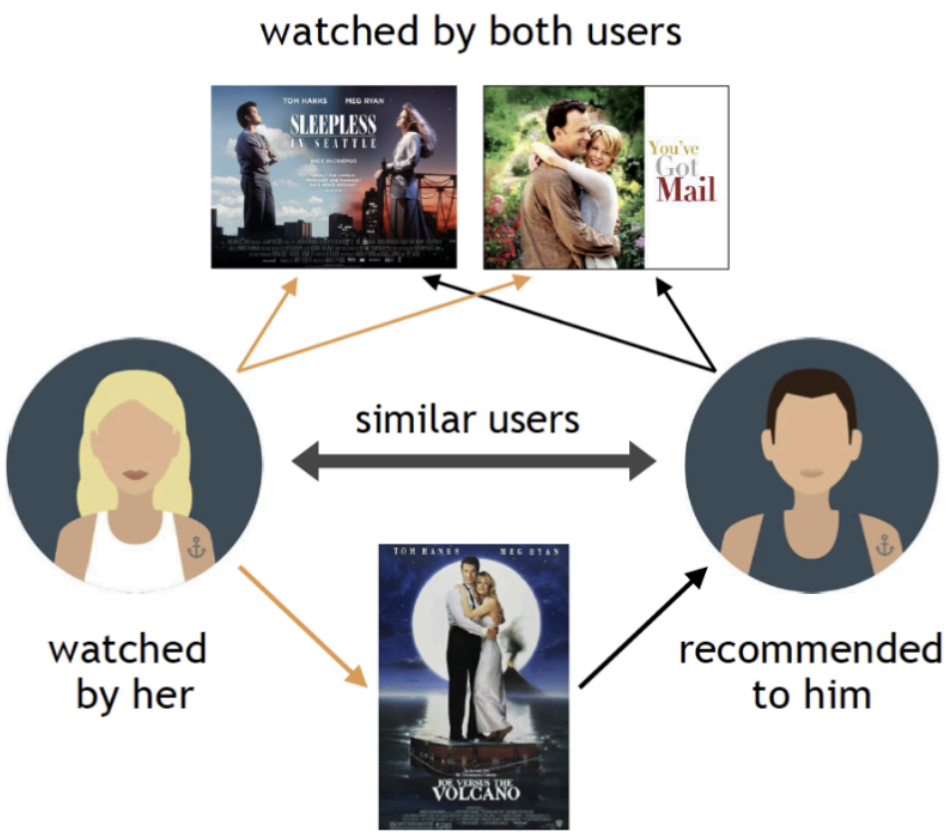
LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

Recommendation paradigms

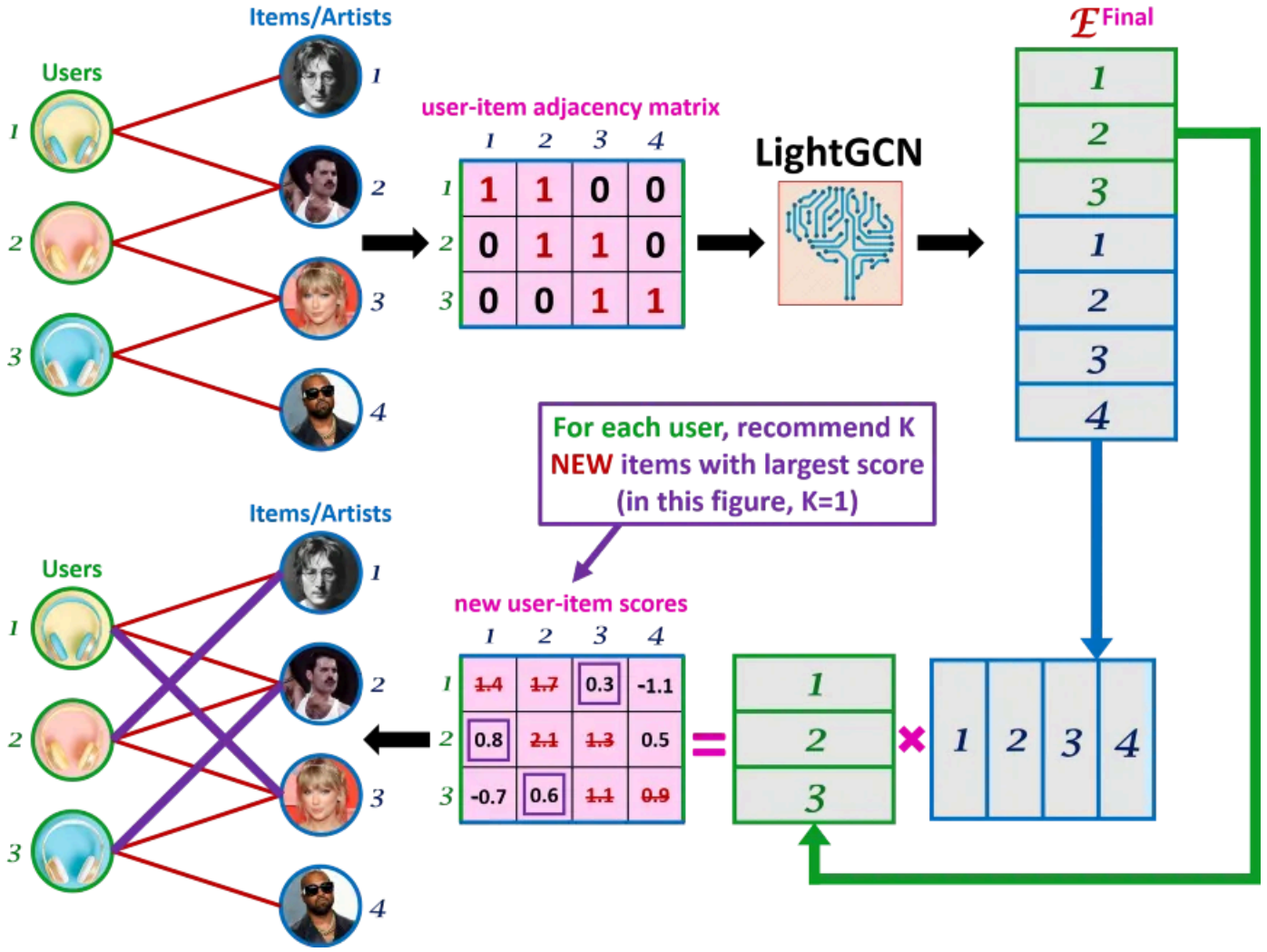
Content-based Filtering



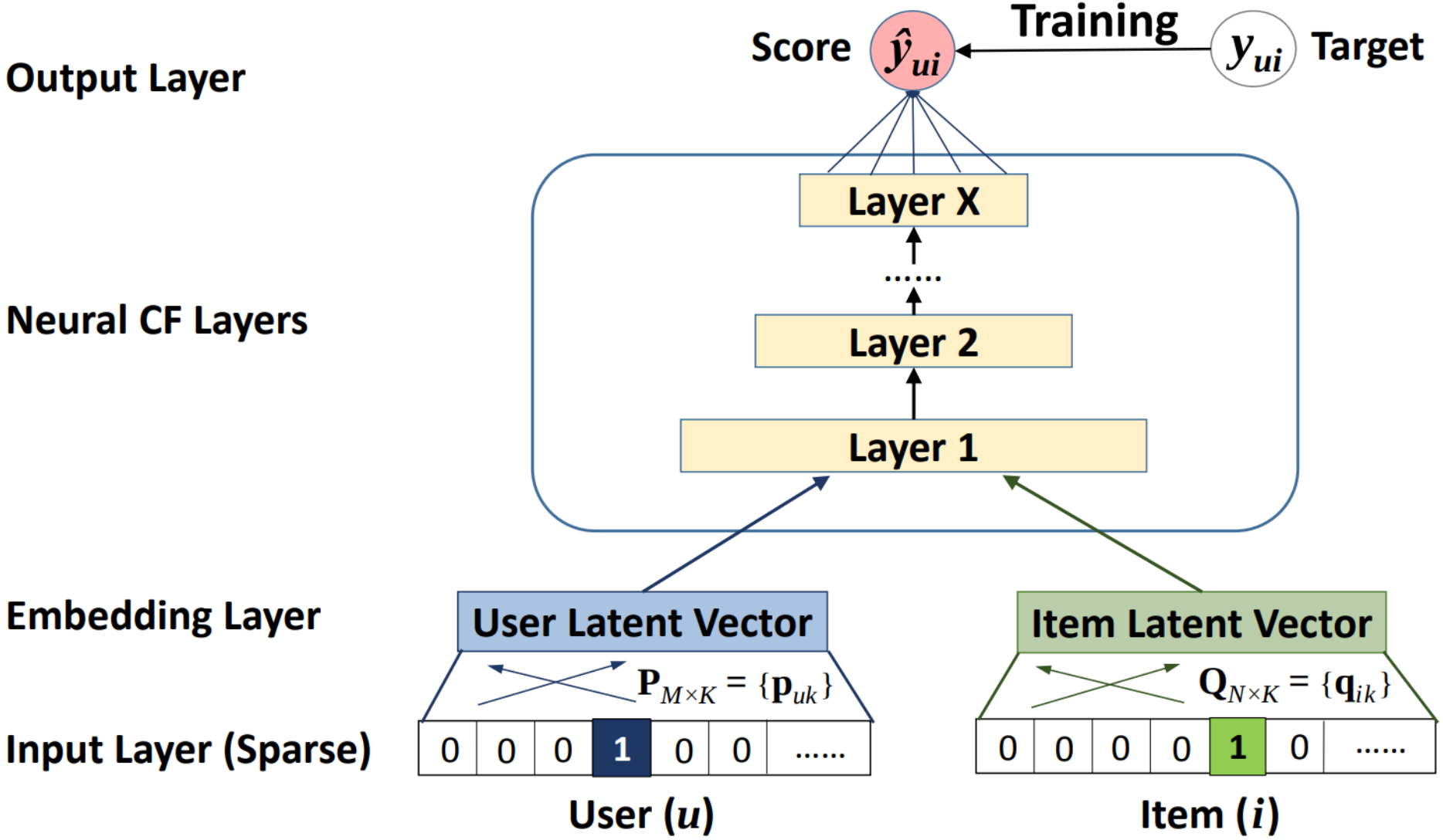
Collaborative Filtering



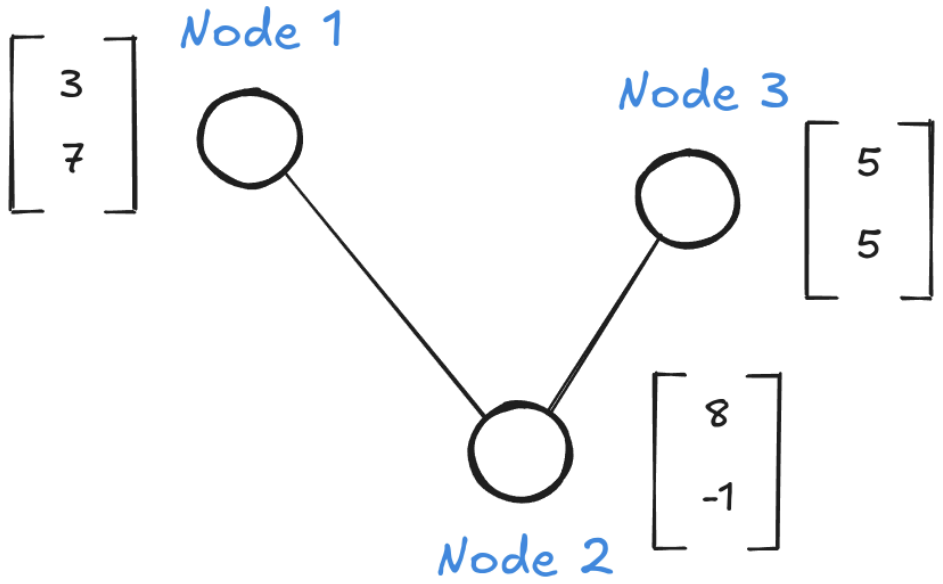
LightGCN role in Collaborative Filtering



Neural network recommendation framework



Graph Convolutional Networks (GCN)



Example of a GCN from literature

$$\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}\big(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\big)\mathbf{W}_1$$

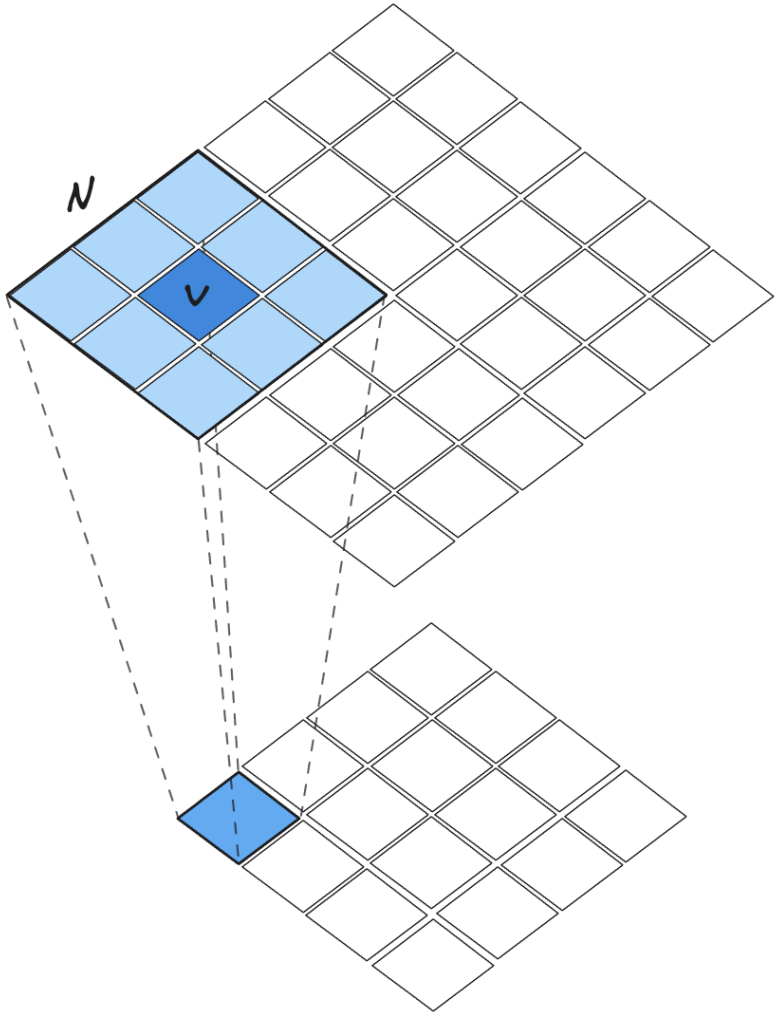
$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

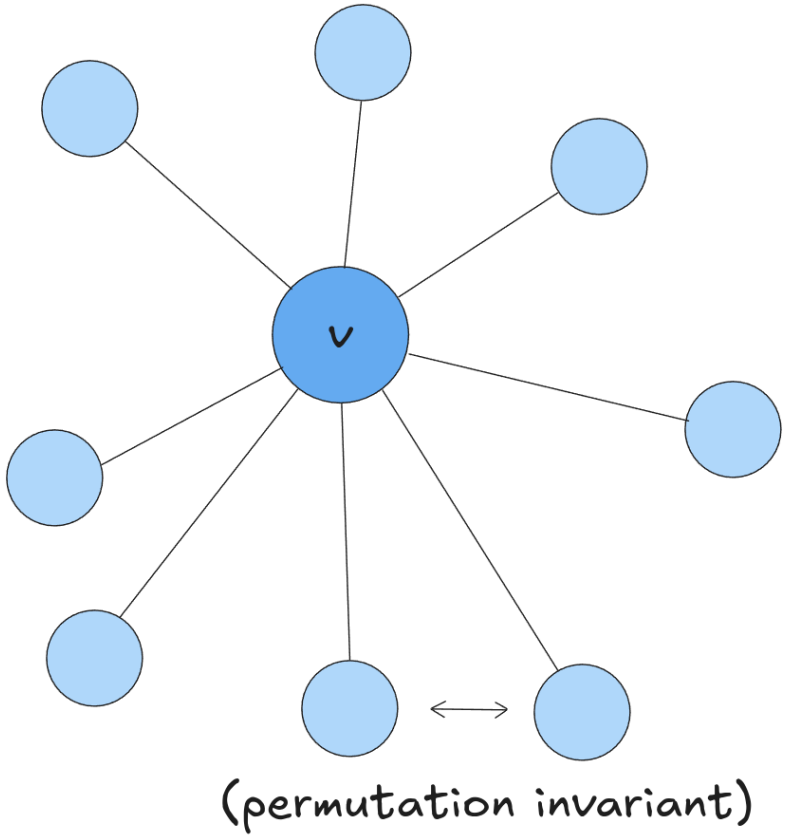
aggregation + combination

$$\begin{matrix}\text{Node 1}\\ \text{Node 2}\\ \text{Node 3}\end{matrix}\begin{bmatrix}1 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 1\end{bmatrix}\begin{bmatrix}3 & 7\\ 8 & -1\\ 5 & 5\end{bmatrix}\begin{bmatrix}\overline{W1}\\ \cdots\\ \overline{Wn}\end{bmatrix}^T = \begin{bmatrix}\overline{Node1} + \overline{Node2}\\ \overline{Node1} + \overline{Node2} + \overline{Node3}\\ \overline{Node2} + \overline{Node3}\end{bmatrix}\begin{bmatrix}\overline{W1}\\ \cdots\\ \overline{Wn}\end{bmatrix}^T$$

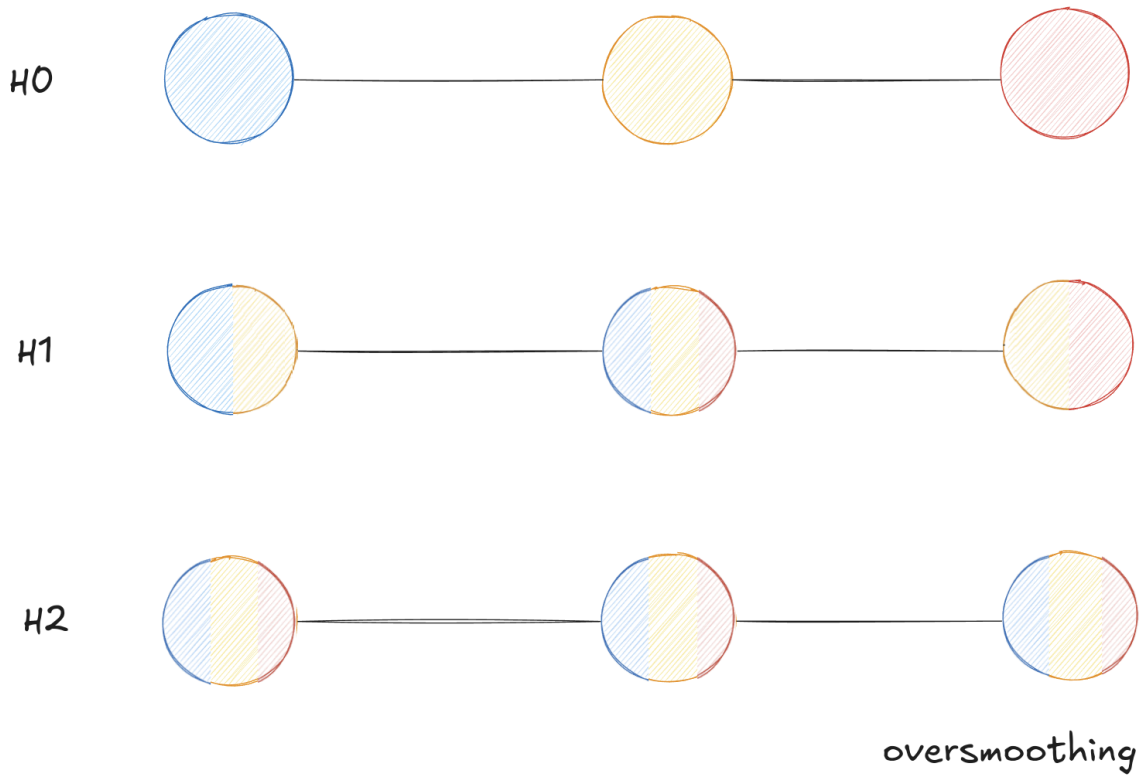
A X W = H W

Graph Convolutions



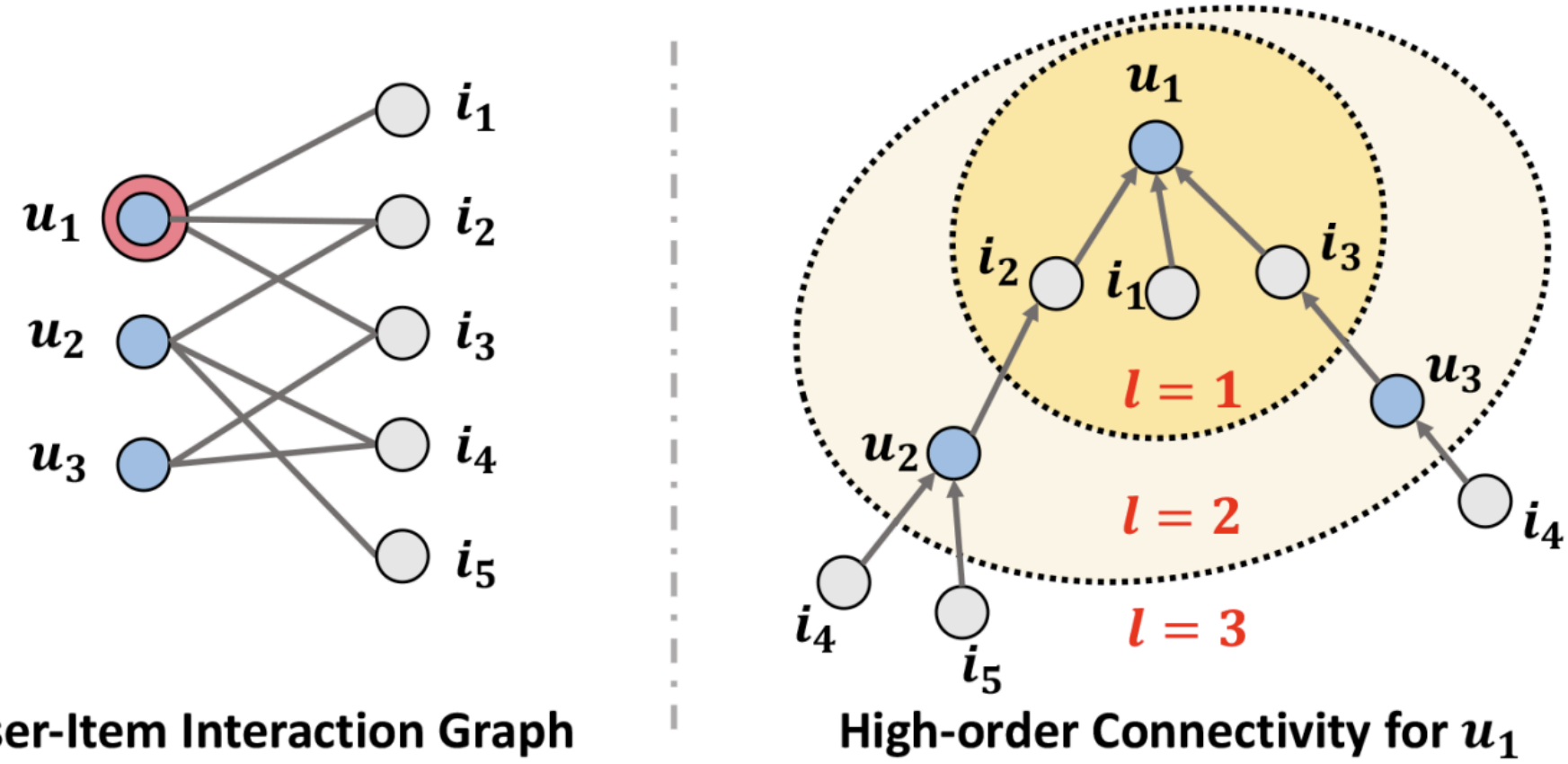


Embeddings



Note that stacking more layers means that information from a given node is able to reach nodes farther away from that node, enabling us to capture the higher-order graph structure.

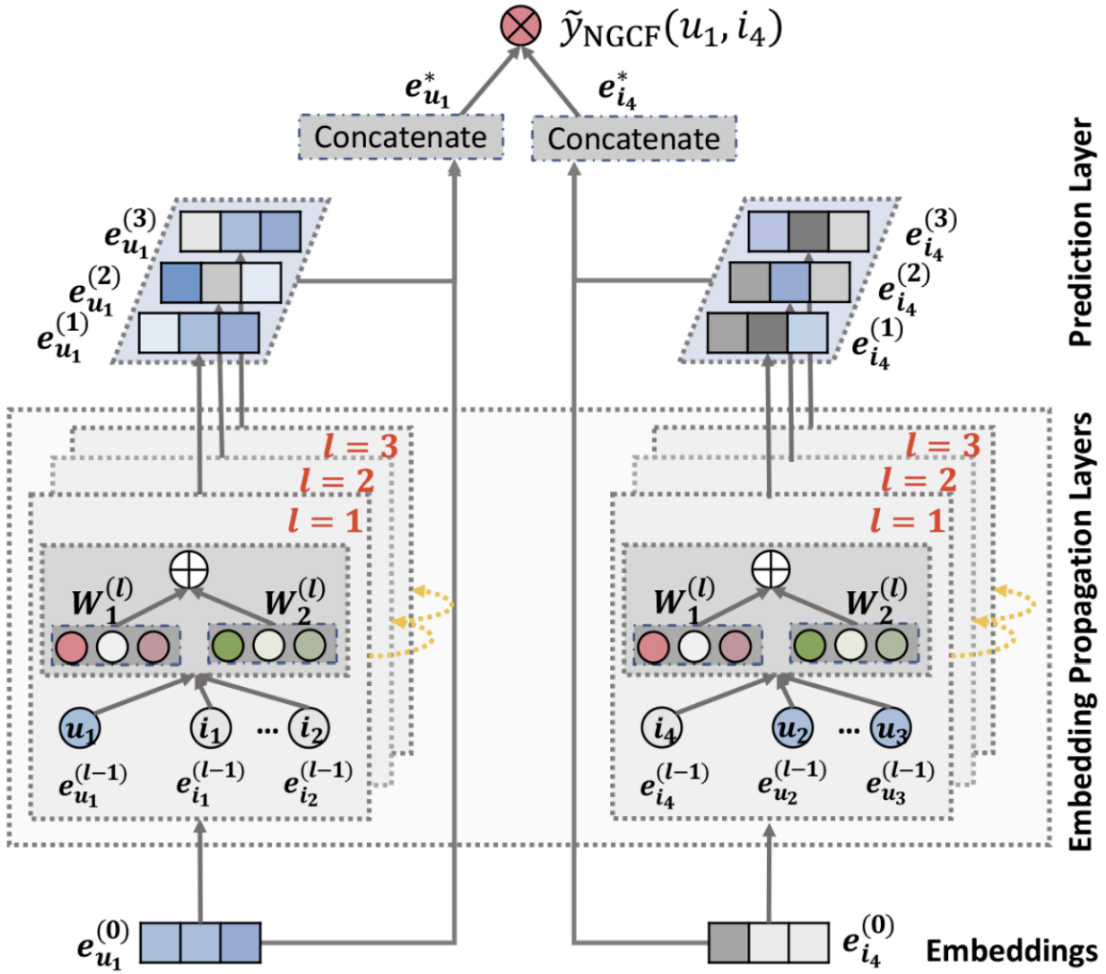
Connectivity



User-Item Interaction Graph **High-order Connectivity for u_1**

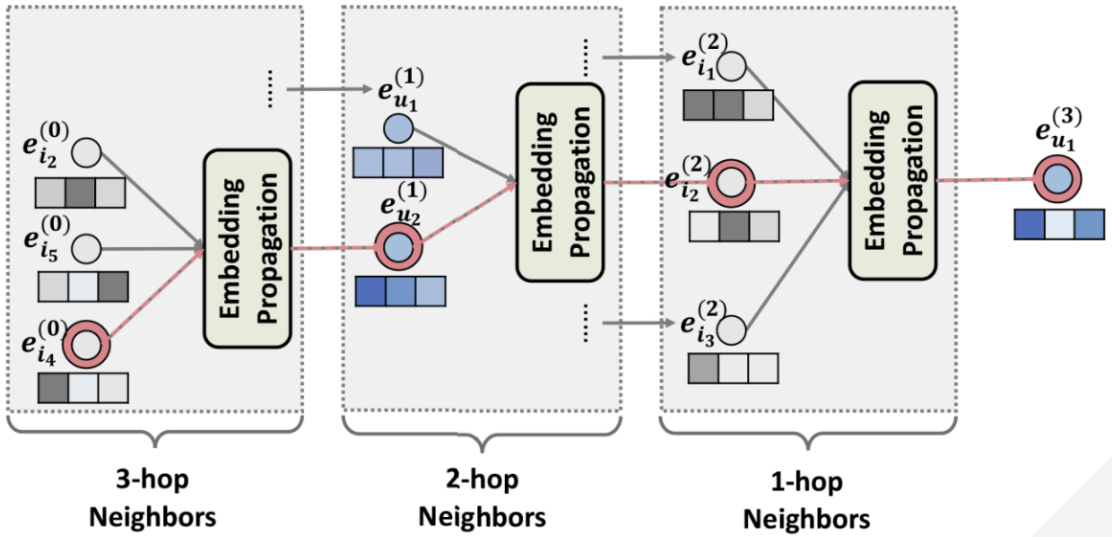
An illustration of the user-item interaction graph and the high-order connectivity. The node u1 is the target user to provide recommendations for.

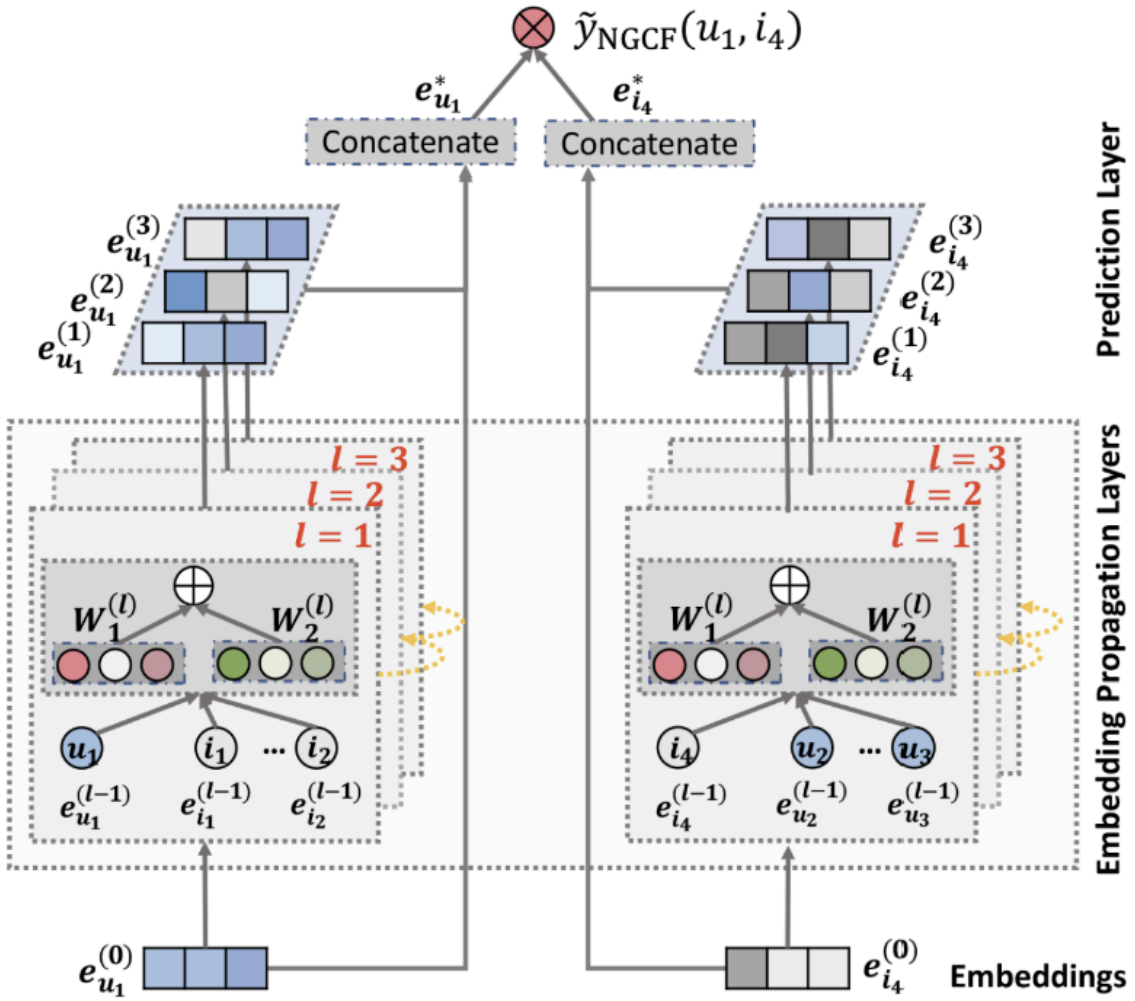
Neural Graph Collaborative Filtering (NGCF)



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

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

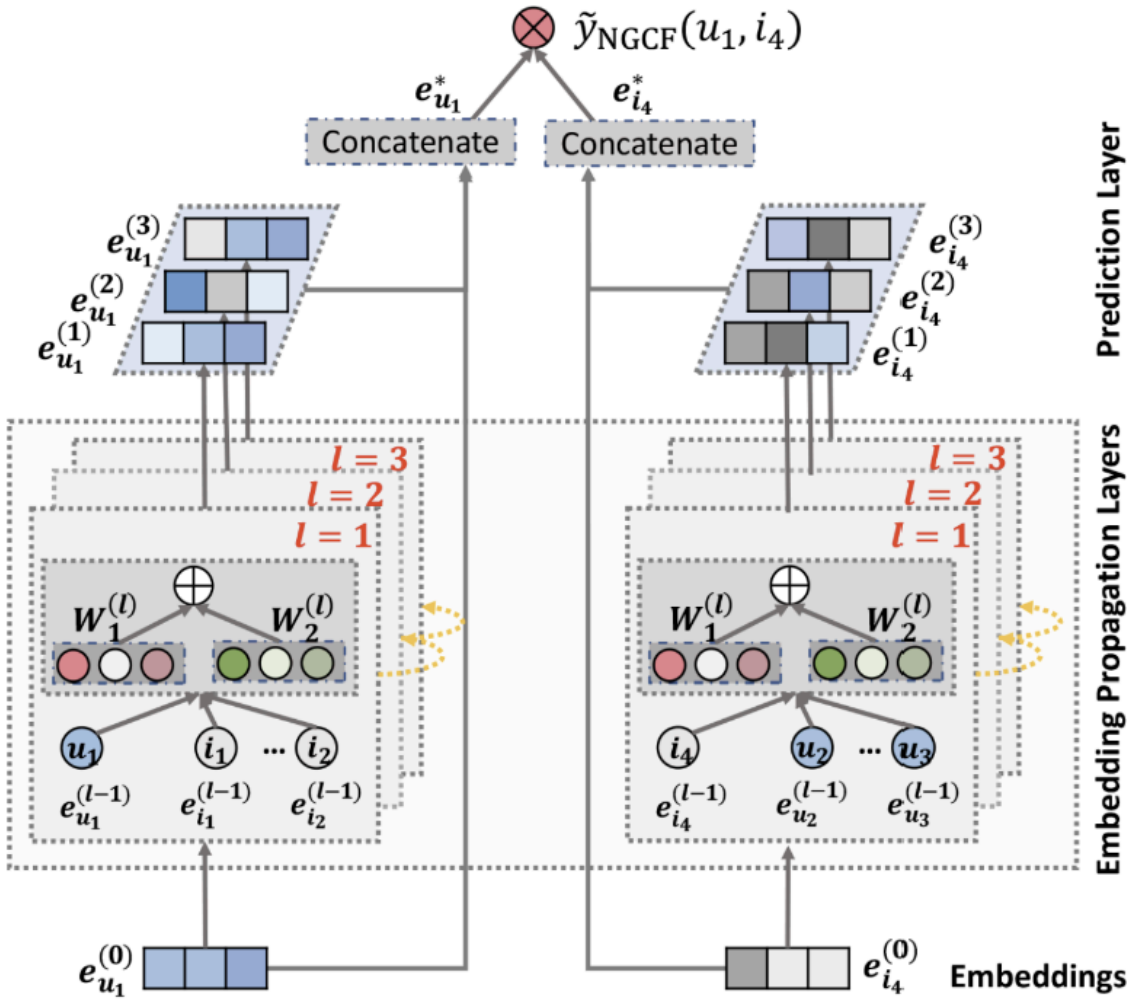




Computation

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

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \text{ and } \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix}$$



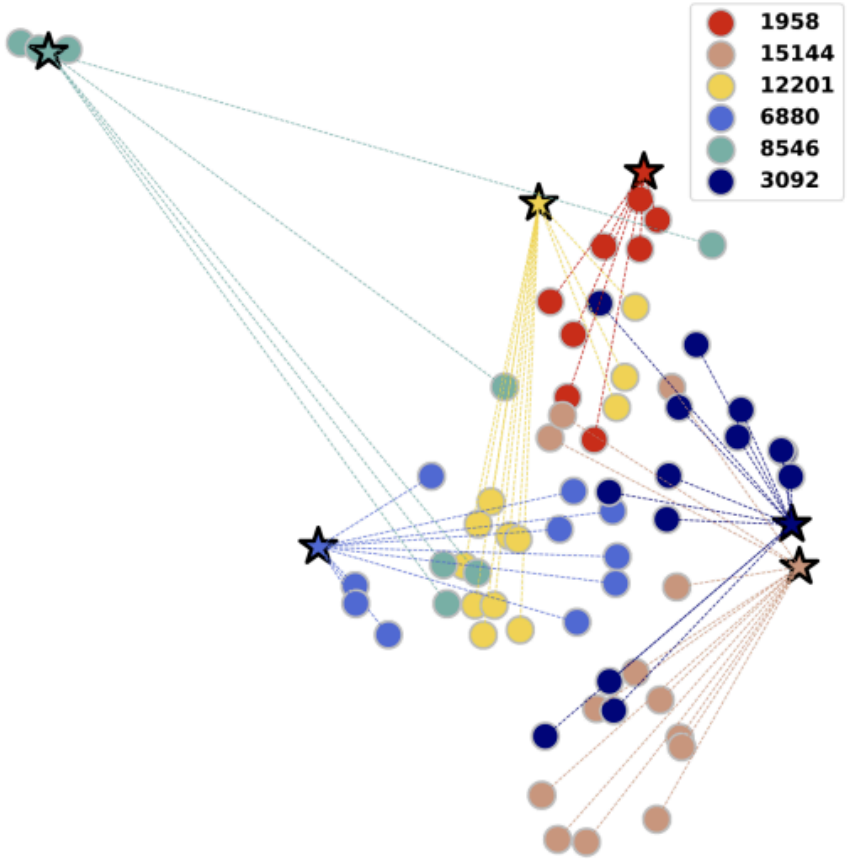
Optimization

$$Loss = \sum_{(u,i,j)\in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2,$$

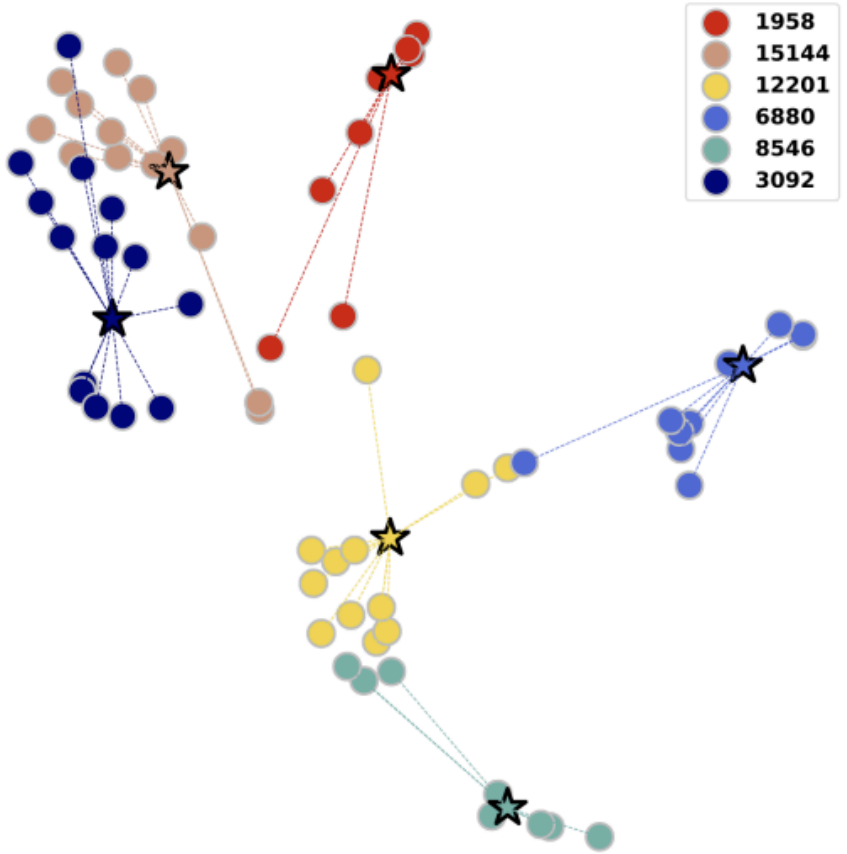
where $O = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$ denotes the pairwise training data, $\mathcal{R}^+$ indicates the observed interactions, and $\mathcal{R}^-$ is the unobserved interactions; $\sigma(\cdot)$ is the sigmoid function; $\Theta = \{\mathbf{E}, \{\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}\}_{l=1}^{L}\}$ denotes all trainable model parameters, and λ controls the L_2 regularization strength to prevent overfitting.

(during training randomly sample a batch of triplets)

Learned representation t-SNE

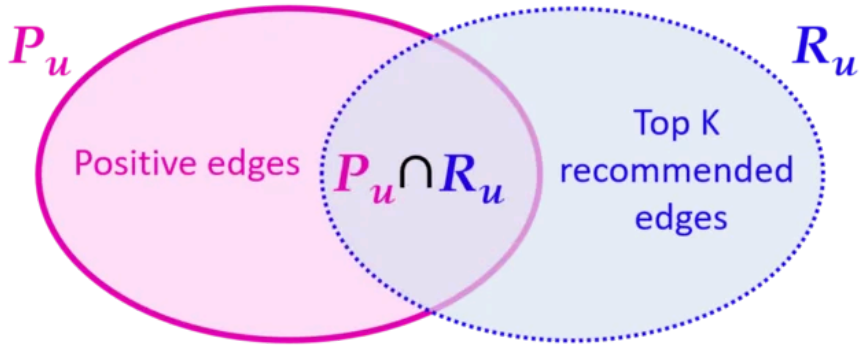


(a) MF (NGCF-0)



(b) NGCF-3

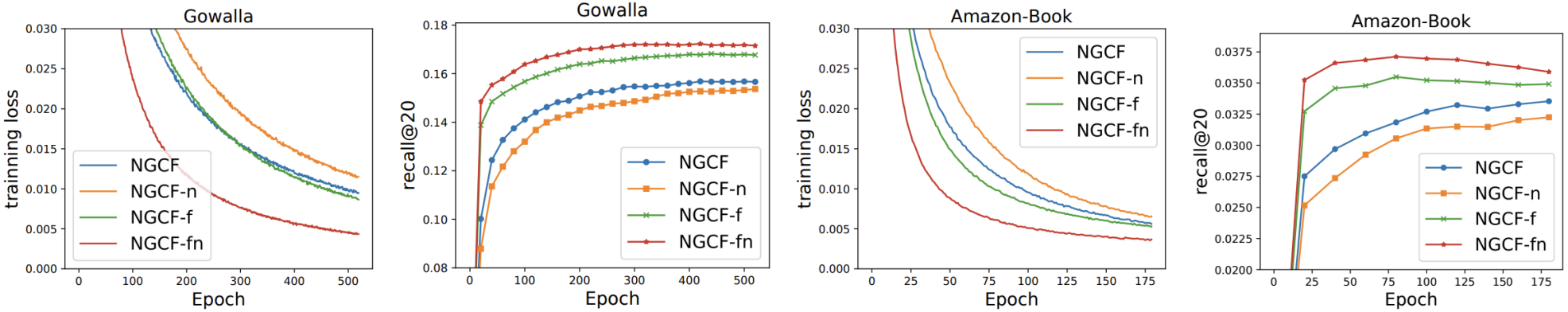
Performance measure



$$\text{Recall@K} = \frac{1}{|\text{Users}|} \sum_{u \in \text{Users}} \left(\frac{P_u \cap R_u}{P_u} \right)$$

Simplifying NGCF

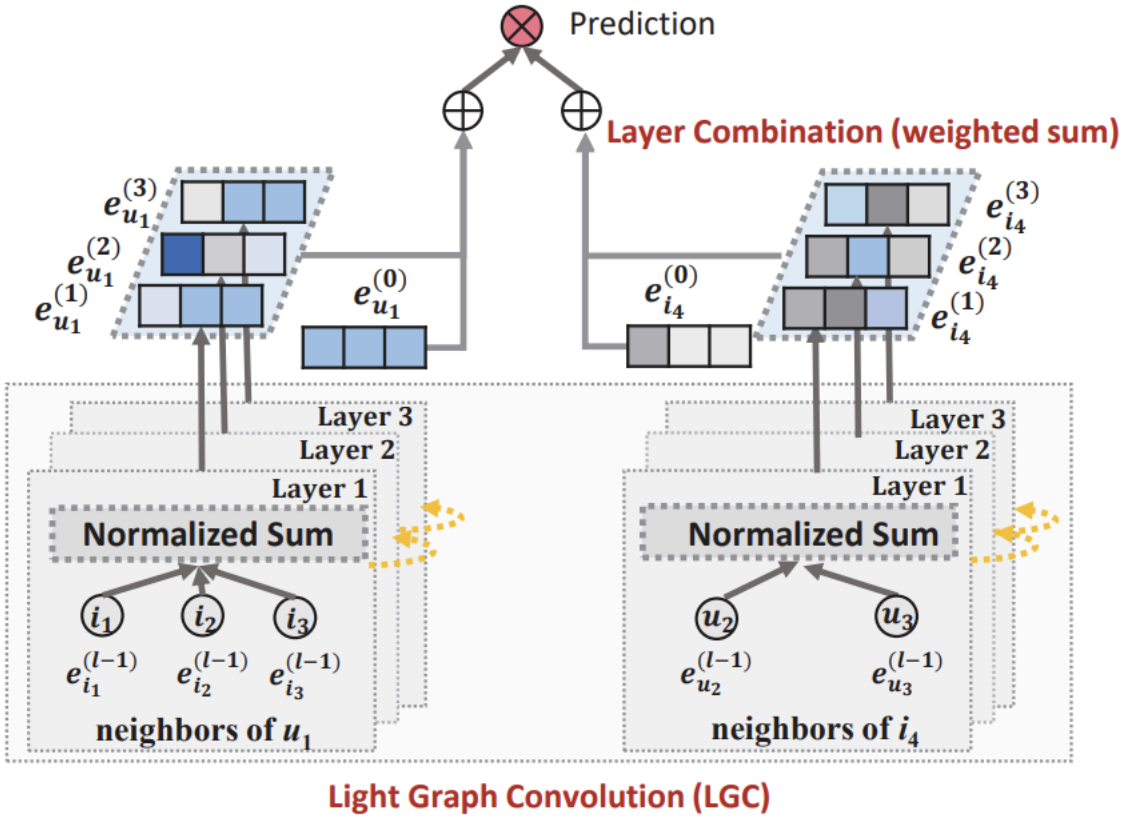
It is argued that eliminating feature extraction is justified, as applying stacked nonlinearities to a structureless input (such as IDs) fails to generate meaningful features and hampers training efficiency.



(a) Training loss on Gowalla (b) Testing recall on Gowalla (c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

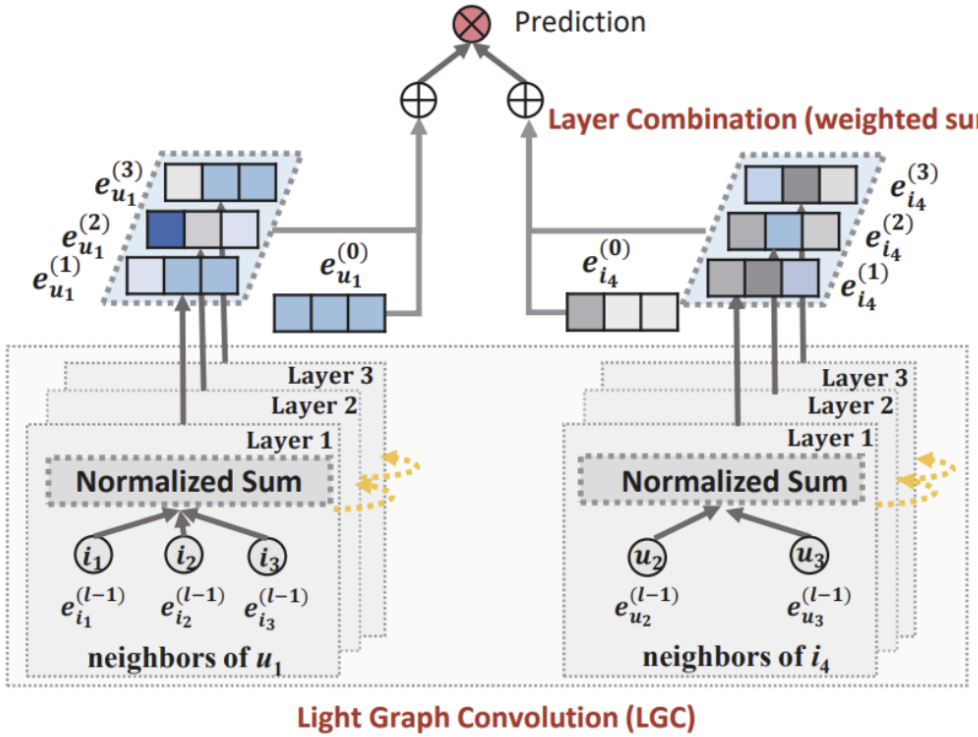
- **NGCF-f** removes the feature transformation matrices W
- **NGCF-n** removes non-linearity σ
- **NGCF-fn** combines the above

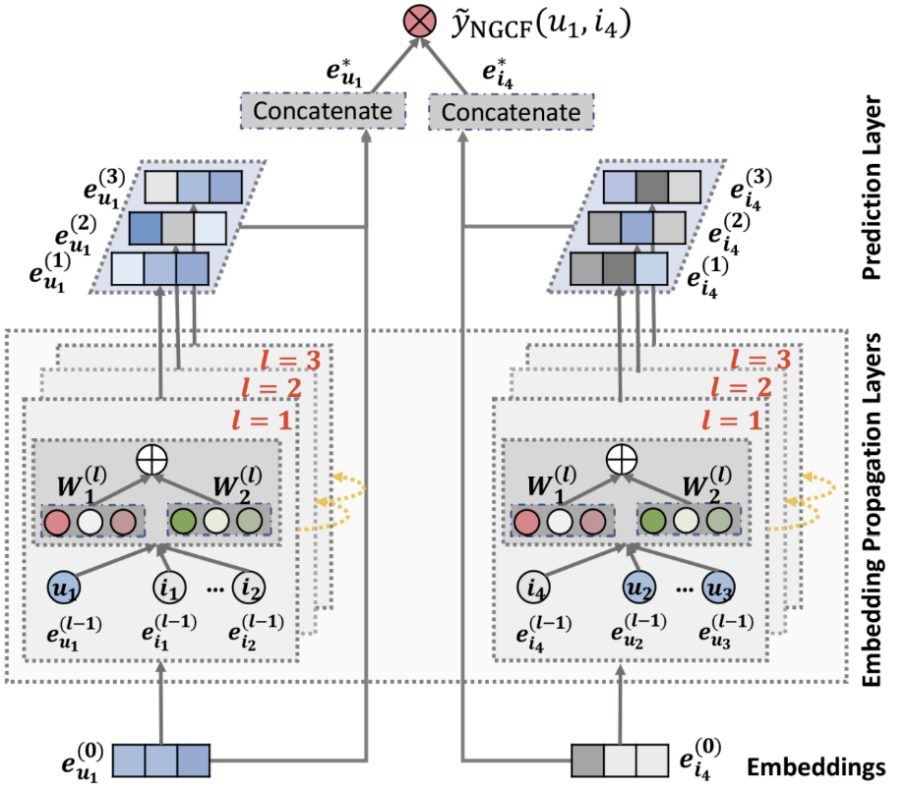
LightGCN (LCN)



Light Graph Convolution (LGC)

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





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

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

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} (\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))\right)$$

$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} (\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}))\right)$$

Performance comparison

Table 3: Performance comparison between NGCF and LightGCN at different layers.

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)

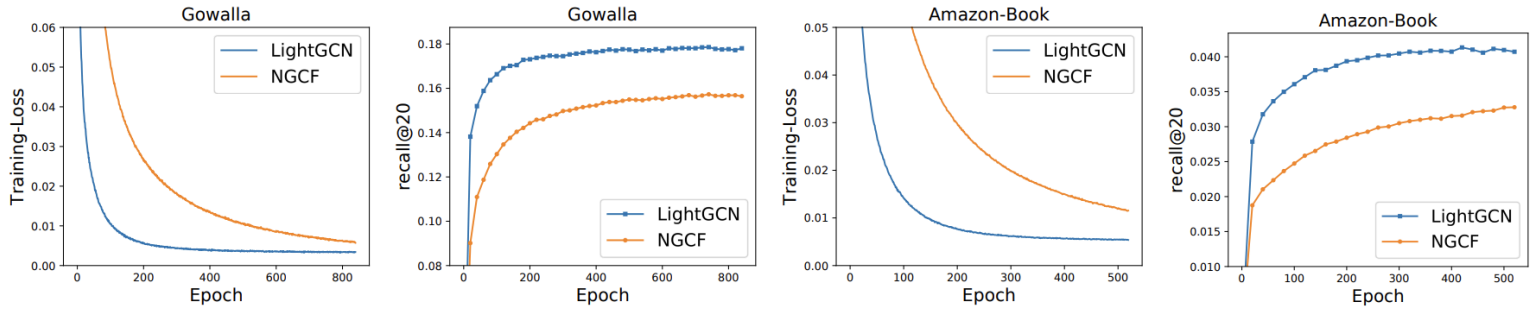


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

Main LightGCN Properties

Relation to SimpleGCN

- **LightGCN**

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

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

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix.

- **SimpleGCN**

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

$$\mathbf{E}^{(K)} = (\mathbf{A} + \mathbf{I})\mathbf{E}^{(K-1)} = (\mathbf{A} + \mathbf{I})^K\mathbf{E}^{(0)}$$

$$= \binom{K}{0}\mathbf{E}^{(0)} + \binom{K}{1}\mathbf{A}\mathbf{E}^{(0)} + \binom{K}{2}\mathbf{A}^2\mathbf{E}^{(0)} + \ldots + \binom{K}{K}\mathbf{A}^K\mathbf{E}^{(0)}$$

The above derivation shows that, inserting self-connection into A and propagating embeddings on it, is essentially equivalent to a weighted sum of the embeddings propagated at each LGC layer.

Relation to APPNP

- **LightGCN**

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

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

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix.

- **APPNP**

$$\mathbf{E}^{(k+1)} = \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(k)}$$

$$\mathbf{E}^{(K)} = \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(K-1)},$$

$$= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + (1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(K-2)}$$

$$= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \beta(1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + (1-\beta)^K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}$$

We see that by setting α accordingly, LightGCN can fully recover the prediction embedding used by APPNP. As such, LightGCN shares the strength of APPNP in combating oversmoothing.

Final remarks on LGC

- Simple to train and implement
- Better generalization ability
- Performance efficient
- Potentially resilient to oversmoothing

End