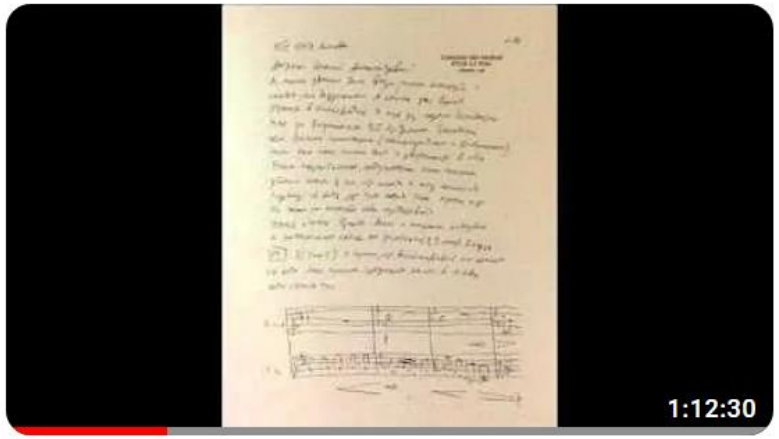
Less is More: Removing Redundancy of Graph Convolutional Networks for Recommendation

Shaowen Peng
Kyoto University, Yoshikawa&ma Lab
tanatosu@foxmail.com

Recommendation



Shostakovich: Symphony No.7 in C major - Mravinsky / Leningrad Philharmonic...

fur bru

5.2万次观看・10年前

Nobel Prize Winners Dr. Katalin Karikó and Dr. Drew Weissman | The Story...

Penn Medicine

7.1万次观看・2周前

“音のない世界”で戦う　姉妹で世界一を目指すデフバドミントン日本代表　2...

福岡TNC ニュース

8924次观看・6天前

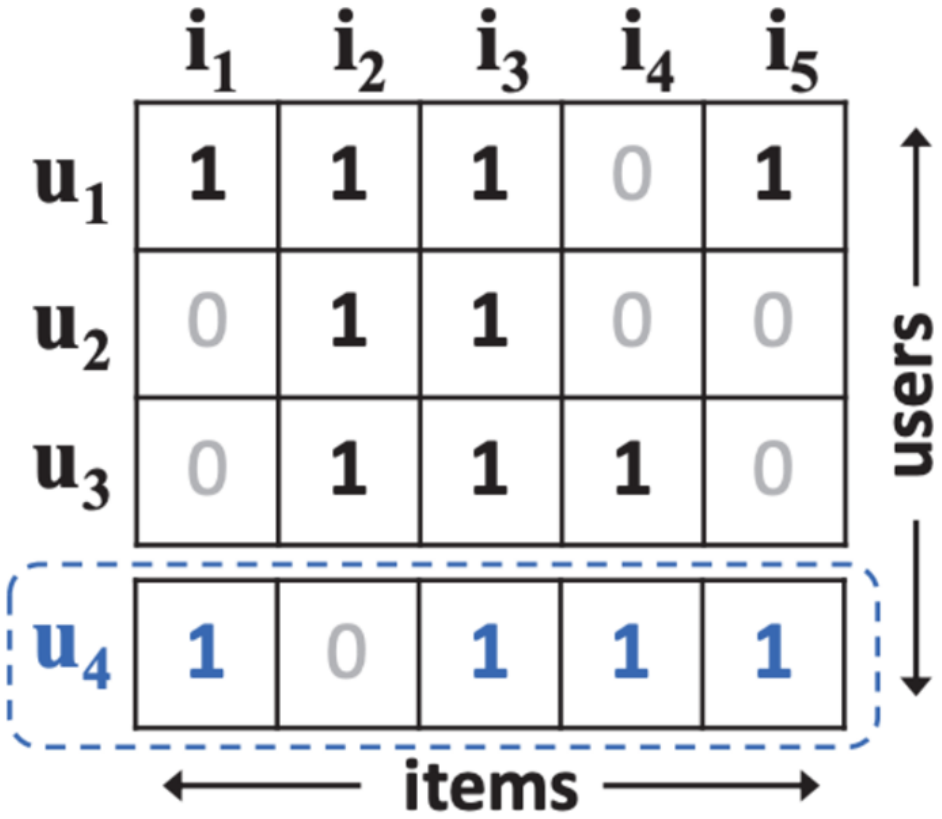
Recommendation

- **Task: Sequential/session-based, Social-based, Image-based, Explainable, Click-rate prediction, Collaborative filtering**

- **Methods: user/item-based, model-based, neural network-based, attention mechanism-based, transformer based, graph-based (graph neural network, GNN)**

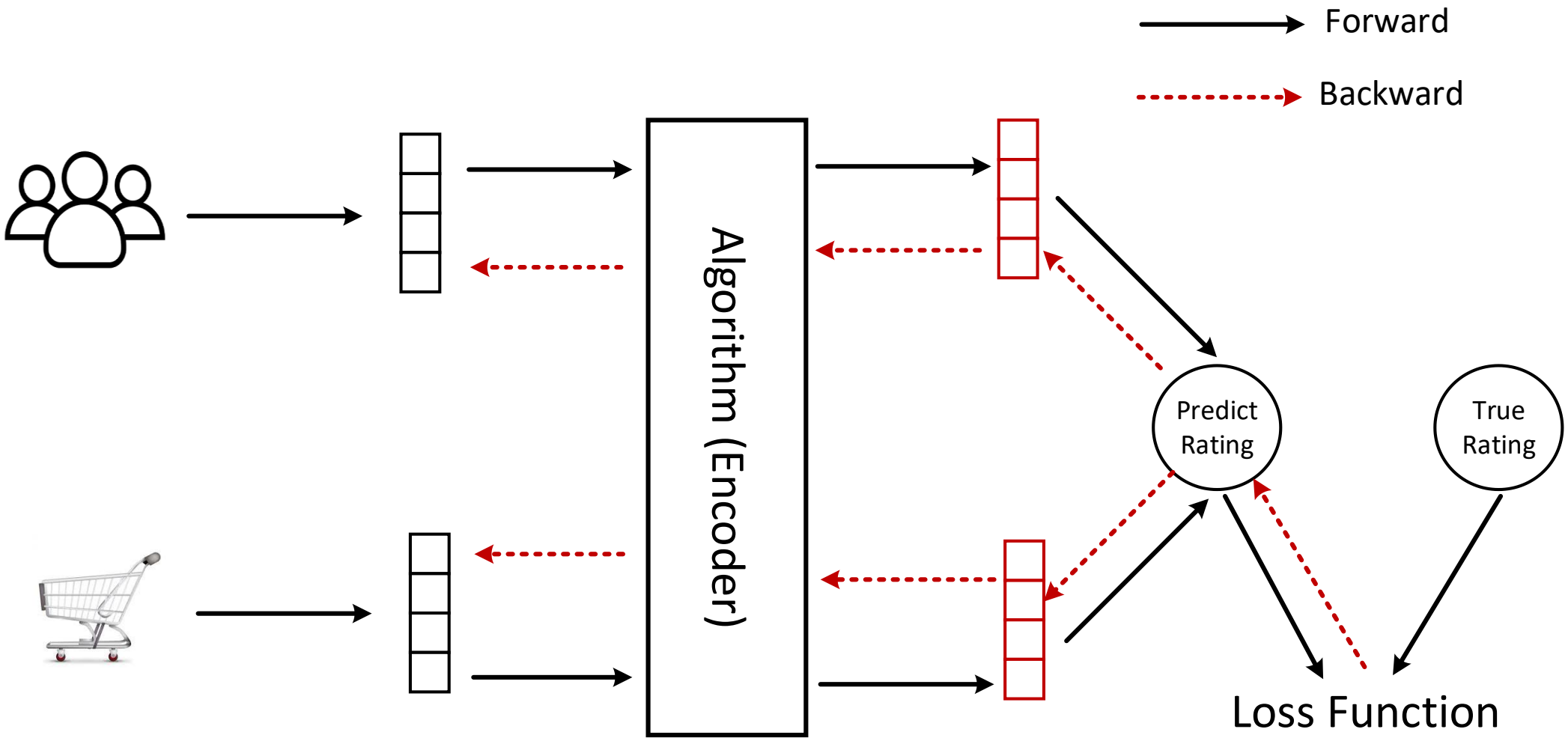
Task Definition

Goal: to predict unobserved interactions based on the interaction matrix $\mathbf{R} \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$. 0 unobserved, 1 observed.

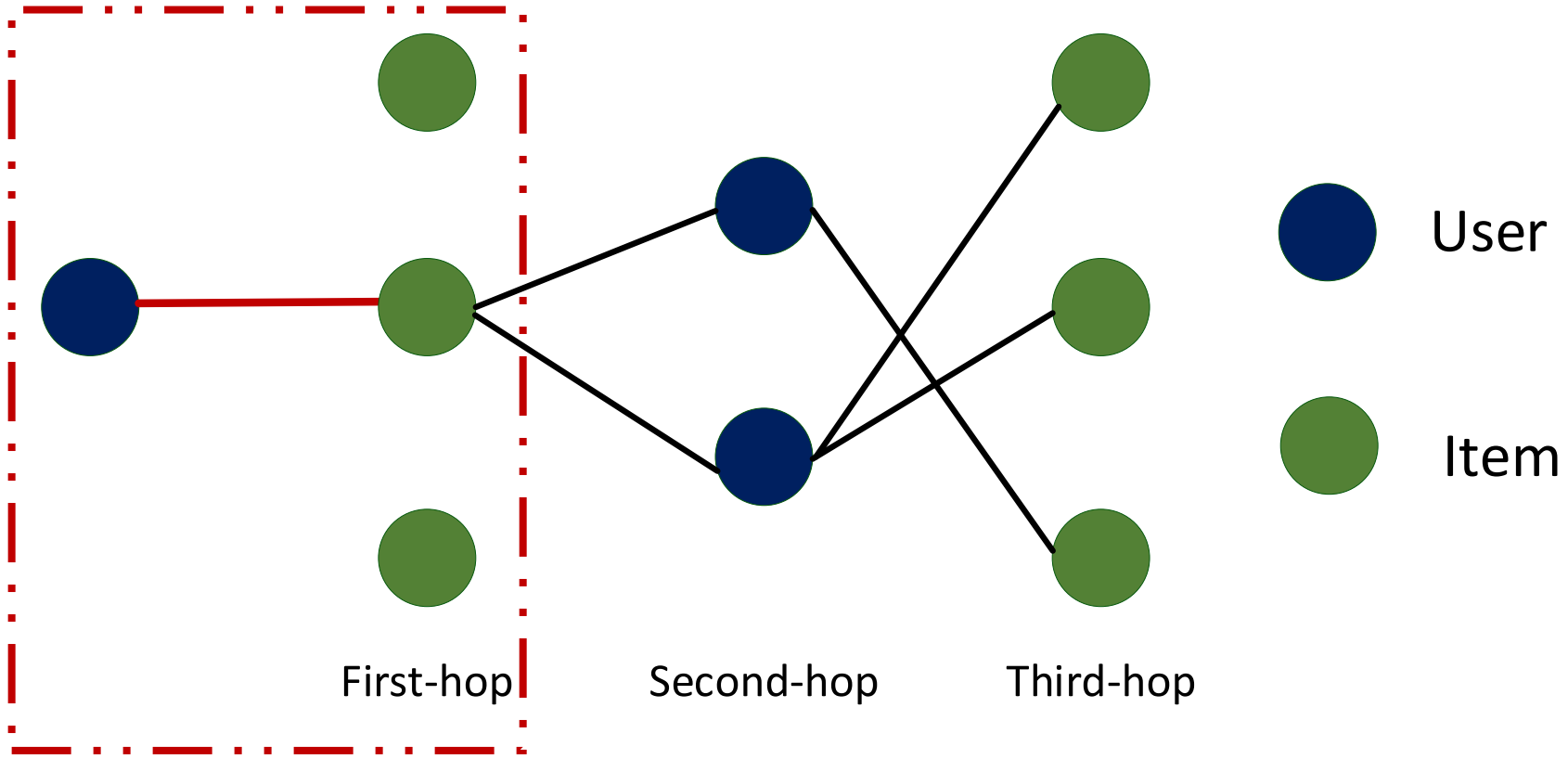


Unlike other tasks, the available data for CF is quite limited and scarce while fundamental.

Learning Paradigm



Why Graph?



How to offer accurate recommendation when users have few interaction data?

Graph Definition

Considering $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$ as a bipartite graph, where users/items are nodes, interactions are edges. The adjacency matrix is defined as:

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

With other side info such as social relation, user-user, item-item can be connected.

A GNN Learning Paradigm for CF

1. Each user/item is characterized as an embedding vector e_u, e_i.

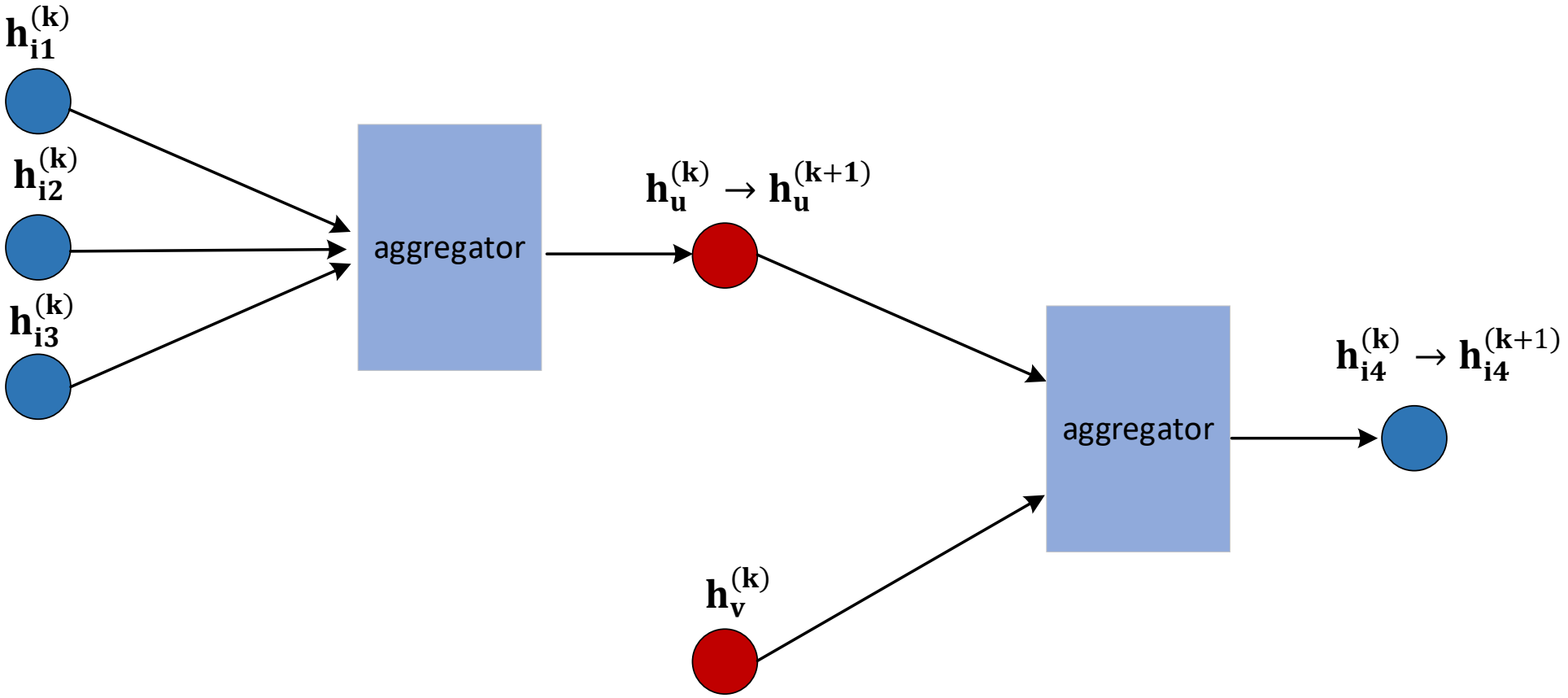
2. $\mathbf{h_u^{(k+1)} = Aggregator(h_i^{(k)} | \ i \ is \ directly \ connected \ to \ u)}$ (repeat)

3. Final embedding $\mathbf{o_u = pooling(h_u^{(k)}, k = \{1, 2, \dots\})}$

4. Predict rating: $\mathbf{o_u^T o_i}$

Aggregator is a function to aggregate message from neighborhood

A GNN Learning Paradigm for CF



The updating is simultaneous across the whole graph

A GNN Learning Paradigm for CF

Most works for CF use Graph Convolutional Networks (GCN), the difference mostly lies in the Aggregator:

(Method): (What algorithm for Aggregator)

Pinsage: Random Walk

KGAT (Knowledge Graph): Attention Mechanism

NGCF, SpectralCF, LightGCN: Adjacency Matrix (GCN)

Graph Convolutional Network (GCN)

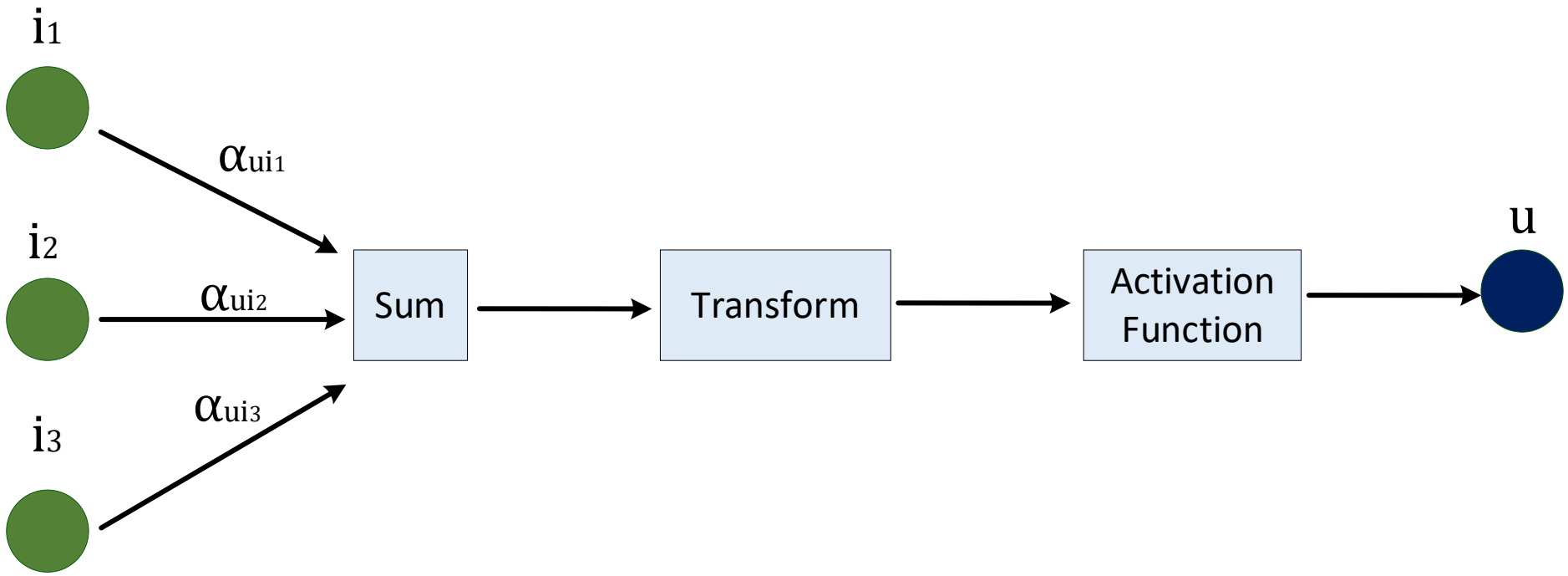
GCN uses normalized adjacency matrix to aggregate neighborhood:

$$\mathbf{H}^{(k+1)} = \sigma\left(\widehat{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}\right)$$

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \hat{\mathbf{R}} \\ \hat{\mathbf{R}}^T & \mathbf{0} \end{bmatrix} \cdot \quad \hat{\mathbf{R}} = \mathbf{D}_U^{-\frac{1}{2}}\mathbf{R}\mathbf{D}_I^{-\frac{1}{2}}$$

The spectral radius can be explosive if we do not normalize adjacency matrix

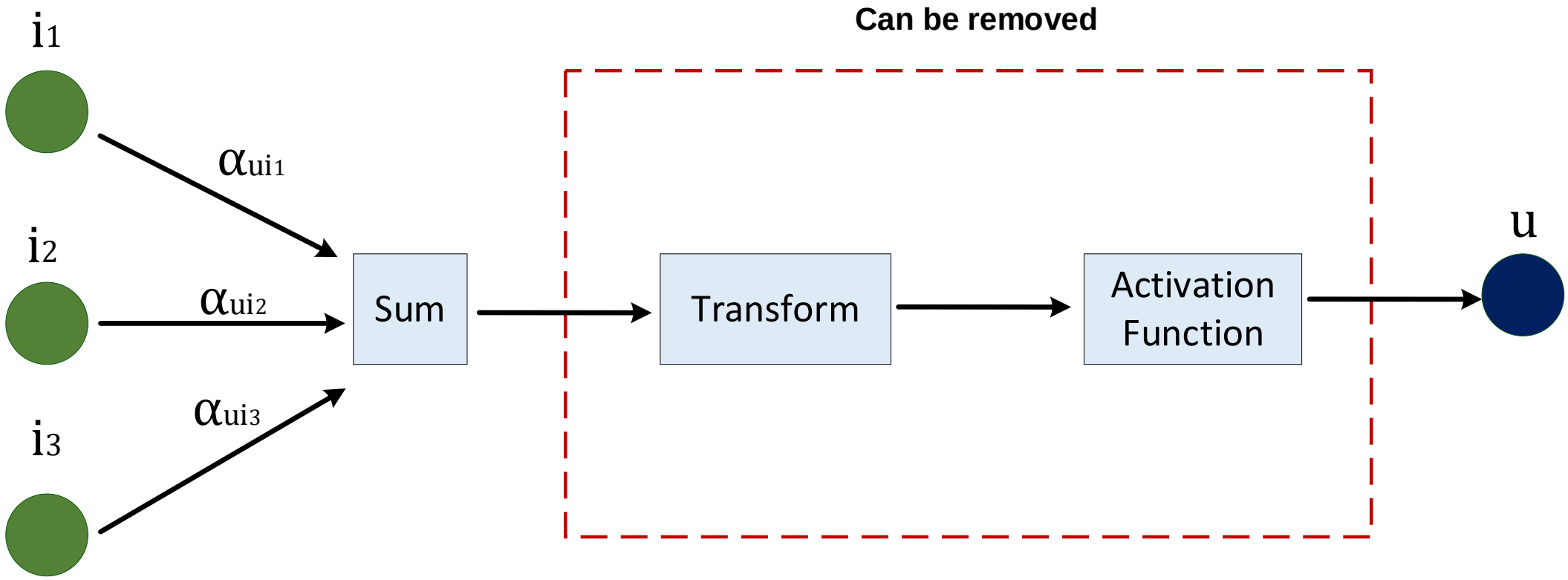
Graph Convolutional Network (GCN)



$$\mathbf{h_u^{(k+1)}} = \sigma\left(\sum_{\mathbf{i}\in\mathcal{N}_u} \boldsymbol{\alpha}_{\mathbf{ui}}\mathbf{h_i^{(k)}}\mathbf{W^{(k+1)}}\right), \mathbf{h_i^{(k+1)}} = \sigma\left(\sum_{\mathbf{u}\in\mathcal{N}_i} \boldsymbol{\alpha}_{\mathbf{iu}}\mathbf{h_u^{(k)}}\mathbf{W^{(k+1)}}\right)$$

$$\boldsymbol{\alpha}_{\mathbf{ui}} = \frac{\mathbf{1}}{\sqrt{\mathbf{d_u}\mathbf{d_i}}}$$ normalized weight, $\mathbf{d_u}$, $\mathbf{d_i}$ are the node degree

LightGCN


Can be removed
i_1
i_2
i_3
α_{ui1}
α_{ui2}
α_{ui3}
Sum
Transform
Activation Function
u

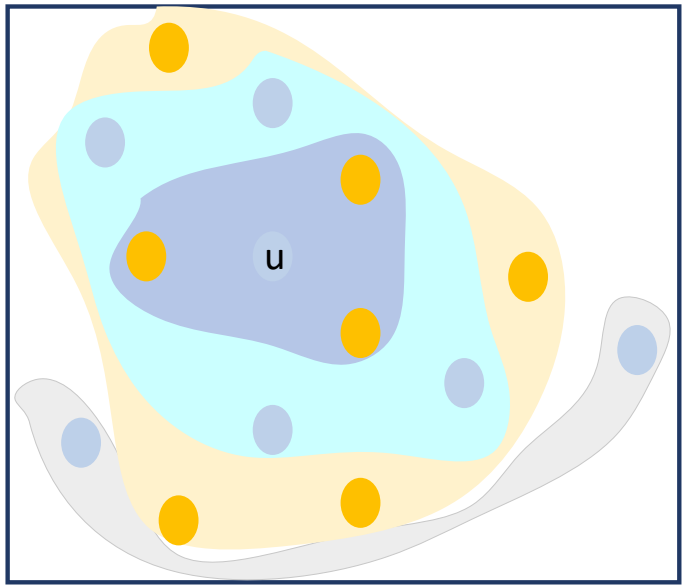
LightGCN

https://dl.acm.org/doi/abs/10.1145/3397271.3401063

Activation function and feature transformation are removed:

$$\mathbf{H}^{(k+1)} = \widehat{\mathbf{A}}\mathbf{H}^{(k)} = \widehat{\mathbf{A}}^{k+1}\mathbf{H}^{(0)}$$

(aggregate k+1 hop neighbors)



LightGCN

Final embeddings are generated as:

$$\mathbf{O} = \sum_{k=0}^{\mathbf{K}} \frac{\mathbf{H}^{(k)}}{\mathbf{K}+\mathbf{1}} = \left(\sum_{k=0}^{\mathbf{K}} \frac{\widehat{\mathbf{A}}^{k}}{\mathbf{K}+\mathbf{1}} \right) \mathbf{E}$$

($\mathbf{E}$ is the initial embedding)

Unsolved Questions

- **Why and how GCN works for recommendation, and why it surpasses traditional and other advanced algorithms?**

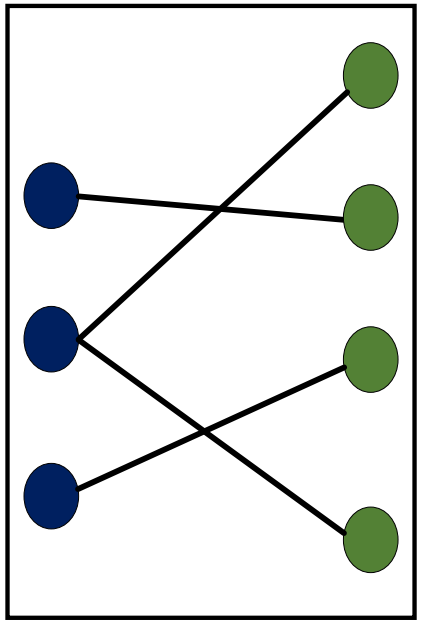
- **Compared with traditional methods, GCN-based methods suffer from high computational cost and poor scalability. Can we design a more efficient GCN learning paradigm?**

Feature Redundancy

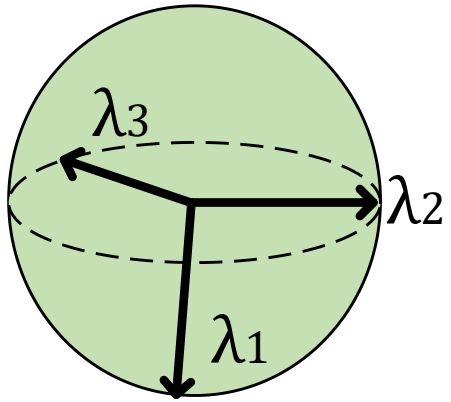
Graph from Spectral Perspective

$\widehat{\mathbf{A}} = \sum_{\mathbf{i}} \boldsymbol{\lambda}_{\mathbf{i}} \mathbf{v}_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{\mathbf{T}}$ (spectral decomposition)

Spatial



Spectral



Graph from Spectral Perspective

$\widehat{\mathbf{A}} = \sum_i \boldsymbol{\lambda_i} \mathbf{v_i} \mathbf{v_i}^T$ (spectral decomposition)

$\mathbf{v_i}$ eigenvector, $\boldsymbol{\lambda_i}$ eigenvalue

Definition: The variation of a graph signal ($\mathbf{v_i}$ here) measures the difference between the signal at each node and its neighborhood:

$$||\mathbf{v_i} - \widehat{\mathbf{A}}\mathbf{v_i}|| = 1 - \boldsymbol{\lambda_i} \in [0,2) \quad \boldsymbol{\lambda_i} \in (-1, 1]$$

- Larger eigenvalue ⟷ Smaller Variation
- **Small variation** implies nodes are similar to their neighbors (Smoothed, Low freq)
- **Large variation** emphasizes the node dissimilarity (Rough, High freq)

Graph from Spectral Perspective

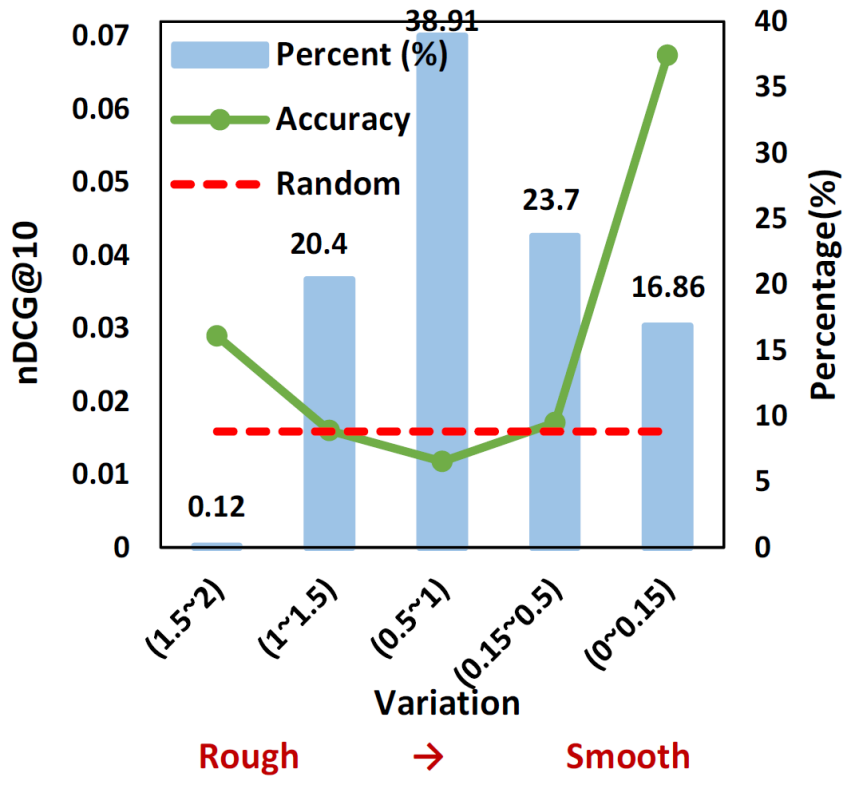
$$\widehat{\mathbf{A}} = \sum_{\mathbf{i}} \boldsymbol{\lambda}_{\mathbf{i}} \mathbf{v}_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{\mathbf{T}}$$ (spectral decomposition)

We are interested in how different spectral features contribute to recommendation

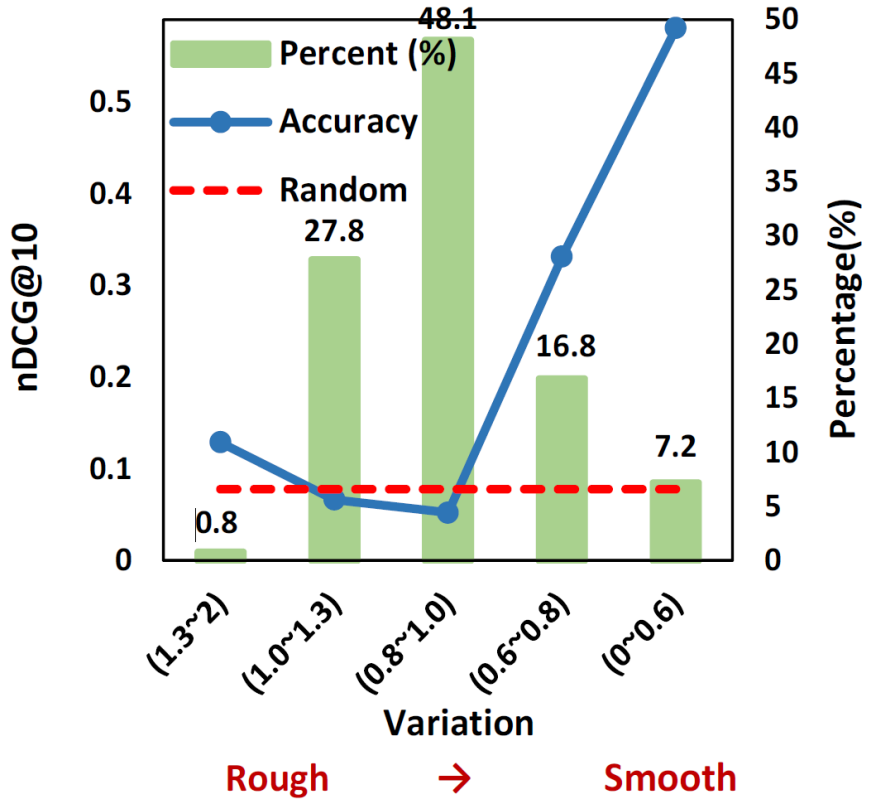
$$\widehat{\mathbf{A}}' = \sum_{\mathbf{i}} \mathbf{M}(\boldsymbol{\lambda}_{\mathbf{i}}) \mathbf{v}_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{\mathbf{T}} \qquad \mathbf{M}(\boldsymbol{\lambda}_{\mathbf{i}}) = \{0, \boldsymbol{\lambda}_{\mathbf{i}}\}$$

For the tested feature $\mathrm{M}(\lambda_i) = \lambda_i$, otherwise $\mathrm{M}(\lambda_i) = 0$

Graph from Spectral Perspective



CiteULike (Sparse)



MovieLens (Dense)

1. **Only a very few features truly contribute to the accuracy.**
2. **The features not either smoothed or rough are useless.**
3. **Smoothed features contribute more to the accuracy than rough ones.**

Graph from Spectral Perspective

What does this mean?

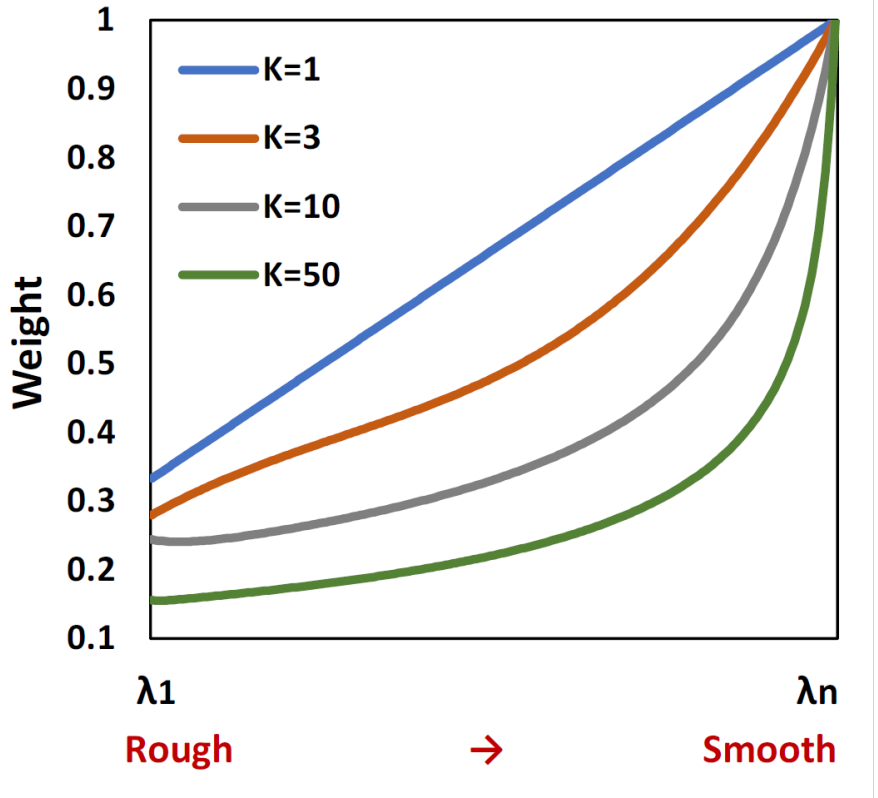
- **User's interactions reflect their preference.**
- **Sometimes user might be interested in the items opposite to user's past interest.**
- **Users are no interested in the items that are not irrelevant (positively or negatively) to their past interest.**

Analysis on LightGCN

$$\mathbf{H}^{(k+1)} = \mathbf{A}\mathbf{H}^{(k)}$$

$$\mathbf{O} = \sum_{k} \frac{\mathbf{H}^{(k)}}{k+1} = \left(\sum_{i} \left(\sum_{k} \frac{\lambda_i^k}{K+1} \right) \mathbf{v}_i \mathbf{v}_i^T \right) \mathbf{E}$$

Weight for v_i



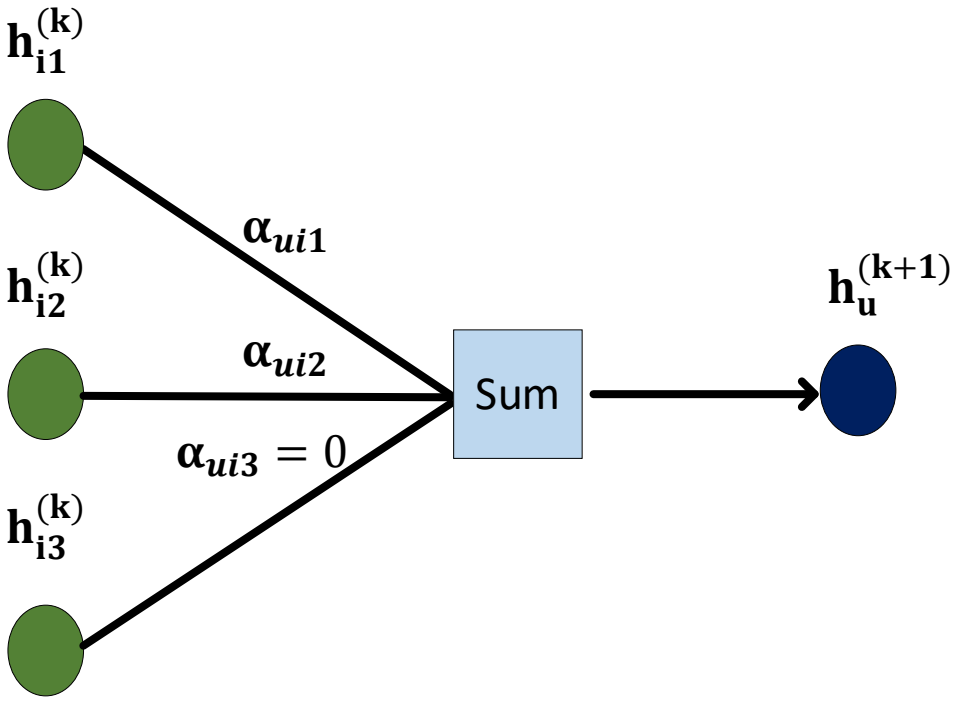
By stacking layers, GCN tends to emphasizes smoothed features and filter out other features.

Spectral Features

- **The importance of different spectral features varies on tasks (recommender system, node classification, anomaly detection, etc.)**
- **The rough features (High frequency) are important when the connected nodes are more different than alike.**
- **“Revisiting graph neural networks: Graph filtering perspective” (Co-authored by Prof. Murata)**

Structure Redundancy

What Makes Representations Different ?



$$\mathbf{O} = \sum_{k=0}^{\mathbf{K}} \frac{\mathbf{H}^{(k)}}{\mathbf{K}+\mathbf{1}} = \left(\sum_{k=0}^{\mathbf{K}} \frac{\hat{\mathbf{A}}^{k}}{\mathbf{L}+\mathbf{1}}\right)\mathbf{E}$$

The difference of different representations lies in $\boldsymbol{\alpha_{ui}}$

Same for all nodes

$$\mathbf{o_k} = \sum_{z \in \mathcal{V}} \alpha_{kz}\,\mathbf{e_z}$$

What Makes Representations Different ?

According to spectral decomposition with V as the eigenmatrix:

$$\alpha_{kz} = \frac{\sum_{l=0}^{L} \widehat{A}_{kz}^{l}}{L+1} = \left(V_{k*} \odot \frac{\sum_{l=0}^{L} \lambda^{l}}{L+1} \right) V_{z*}^{T}$$

V_{k*} **is k-th row of V,** $\odot$ **stands for the element-wise multiplication**

What Makes Representations Different ?

$$\mathbf{o_u} = \left(\mathbf{V_{u*}} \odot \frac{\sum_{l=0}^{L} \lambda^l}{L+1} \right) \mathbf{V^T E}$$

$$\mathbf{o_i} = \left(\mathbf{V_{i*}} \odot \frac{\sum_{l=0}^{L} \lambda^l}{L+1} \right) \mathbf{V^T E}$$

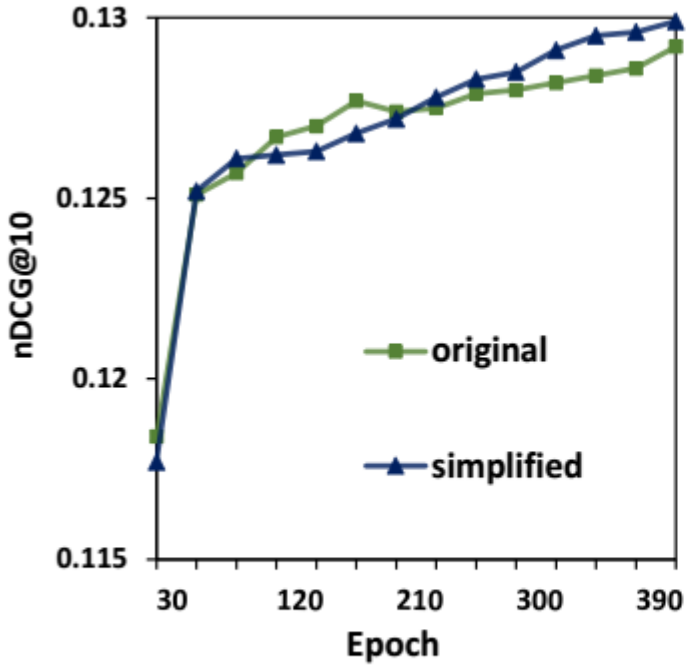
Common for all nodes

Weighted Spectral Features

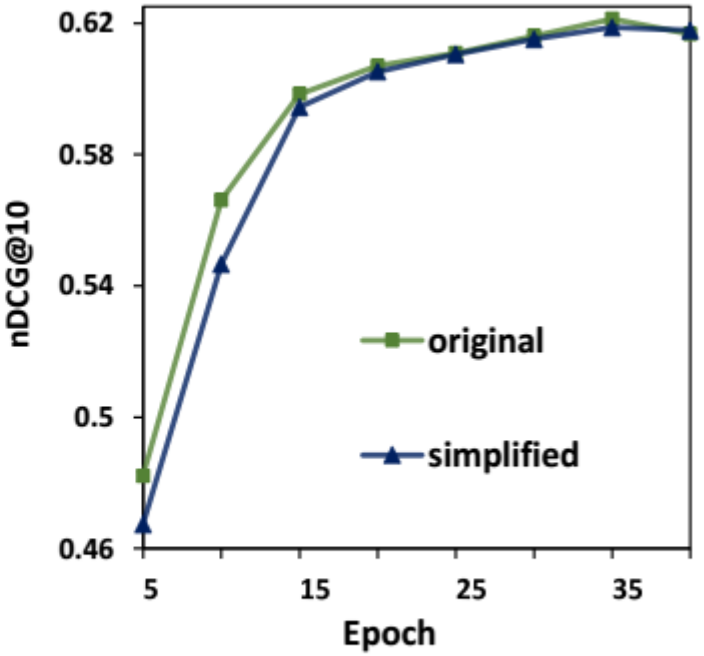
What Makes Representations Different ?

Original: LightGCN

Simplified: Replacing the term V^TE with a weight matrix



(a) CiteULike



(b) ML-100K

What Makes Representations Different ?

$$\mathbf{o_u} = \left(\mathbf{V_{u*}} \odot \frac{\sum_{\mathbf{l=0}}^{\mathbf{L}} \boldsymbol{\lambda}^{\mathbf{l}}}{\mathbf{L+1}} \right) \mathbf{W}$$

$$\mathbf{o_i} = \left(\mathbf{V_{i*}} \odot \frac{\sum_{\mathbf{l=0}}^{\mathbf{L}} \boldsymbol{\lambda}^{\mathbf{l}}}{\mathbf{L+1}} \right) \mathbf{W}$$

Without W, unparameterized model, optimization not required

The Simplified model does not requires neighborhood aggregation, it is equivalent to a linear model

Summary

Summarizing the findings from Feature and Structure redundancy, we can simplify GCN as:

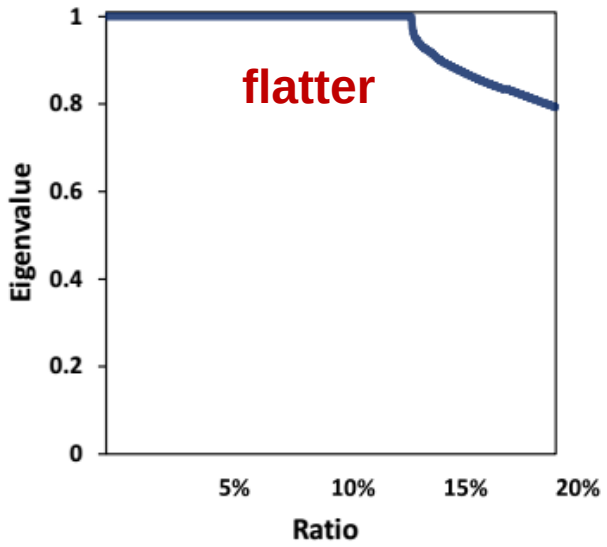
$$\mathbf{O_U} = \mathbf{P}^{(\mathbf{K})}\mathbf{diag}\left(\mathbf{\Delta}(\boldsymbol{\sigma}_\mathbf{k})\right)\mathbf{W}$$
$$\mathbf{O_I} = \mathbf{Q}^{(\mathbf{K})}\mathbf{diag}\left(\mathbf{\Delta}(\boldsymbol{\sigma}_\mathbf{k})\right)\mathbf{W}$$

It has the following three components:

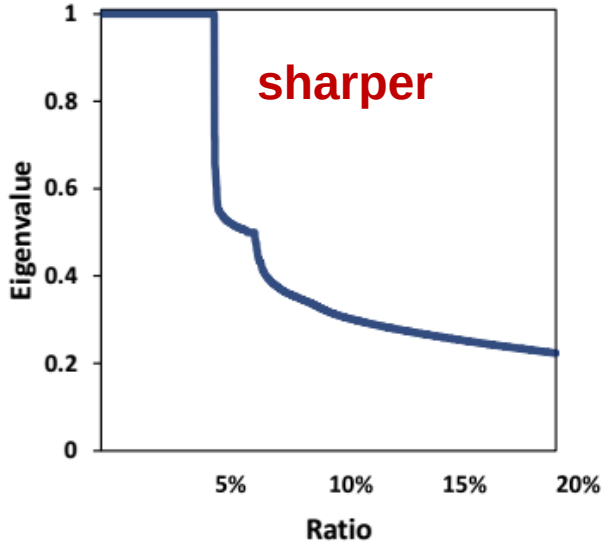
- Stacked top-K smoothest left and right singular vectors of $\widehat{\mathbf{R}}$: $\mathbf{P}^{(\mathbf{K})} \in \mathbf{R}^{|\mathbb{U}|\times\mathbf{K}}$ and $\mathbf{Q}^{(\mathbf{K})} \in \mathbf{R}^{|\mathbb{I}|\times\mathbf{K}}$.
- GCNs use a polynomial to weight the spectral features. Here, we abstract it as a nonparametric function $\Delta(\cdot)$
- A feature transformation $\mathbf{W} \in \mathbf{R}^{\mathbf{K}\times\mathbf{d}}$, $K \ll \min(|\mathcal{U}|, |\mathcal{I}|)$ (**optional**)

Distribution Redundancy

Spectral Distribution Matters



(a) Top 20% eigenvalue distribution on CiteULike.



(b) Top 20% eigenvalue distribution on ML-1M.

We observe that a the graph with a flatter (sharper) spectral distribution (the spectral value drops more slowly (quickly)) requires more (fewer) features.

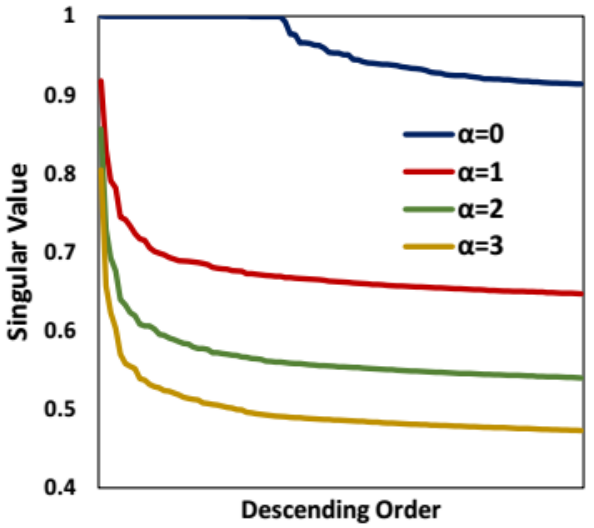
How to Reduce the Spectral Features?

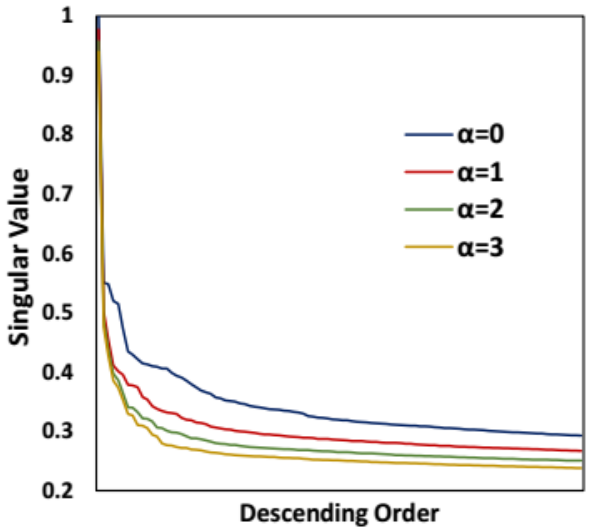
$$\widehat{\mathbf{A}}_{\mathbf{ui}} = \frac{\mathbf{1}}{\sqrt{\mathbf{d_u d_i}}} \longrightarrow \overline{\mathbf{A}}_{\mathbf{ui}} = \mathbf{w}(\mathbf{d_u})\mathbf{w}(\mathbf{d_i})$$

$\mathbf{w}(\mathbf{d_u}) = \frac{\mathbf{1}}{\sqrt{\mathbf{d_u}}}$ in the original setting

The higher weights over the high-degree-nodes tend to sharpen the distribution

Sharpen the Distribution





(e) Top 100 singular values with varying α on CiteULike.

(f) Top 100 singular values with varying α on ML-1M..

$$\mathbf{w}(\mathbf{d_u}) = \frac{\mathbf{1}}{\sqrt{\mathbf{d_u} + \boldsymbol{\alpha}}}$$

Adversarial Training

$$\mathbf{O_U} = \mathcal{N}\left(\mathbf{P}^{(\mathbf{K})}\mathbf{diag}\left(\boldsymbol{\Delta}(\boldsymbol{\sigma}_\mathbf{k})\right), \boldsymbol{\mu}\right)\mathbf{W}$$

$$\mathbf{O_I} = \mathcal{N}\left(\mathbf{Q}^{(\mathbf{K})}\mathbf{diag}\left(\boldsymbol{\Delta}(\boldsymbol{\sigma}_\mathbf{k})\right), \boldsymbol{\mu}\right)\mathbf{W}$$

Training Objectives

$$\mathcal{L}_{main} = \sum_{u \in \mathcal{U}} \sum_{(u,i^+) \in \mathbf{R}^+, (u,i^-) \notin \mathbf{R}^+} \ln \sigma \left(\mathbf{o_u^T o_{i^+}} - \mathbf{o_u^T o_{i^-}} \right)$$

$$\mathcal{L}_{user} = \sum_{u \in \mathcal{U}} \sum_{(u,u^+) \in, (u,u^-) \notin \mathcal{E}^{\mathcal{L}}_{\mathcal{A}_\mathcal{U}}} \ln \sigma \left(\mathbf{o_u^T o_{u^+}} - \mathbf{o_u^T o_{u^-}} \right)$$

user-user relations

$$\mathcal{L}_{item} = \sum_{i \in \mathcal{I}} \sum_{(i,i^+) \in, (i,i^-) \notin \mathcal{E}^{\mathcal{L}}_{\mathcal{A}_\mathcal{I}}} \ln \sigma \left(\mathbf{o_i^T o_{i^+}} - \mathbf{o_i^T o_{i^-}} \right)$$

Item-item relations

$$\mathcal{L} = \mathcal{L}_{main} + \delta \mathcal{L}_{user} + \zeta \mathcal{L}_{item} + \gamma |\Theta|_2^2$$

Experiments

Accuracy

Datasets	Methods	nDCG@10	nDCG@20	Recall@10	Recall@20
CiteULike	LightGCN	0.0751	0.0710	0.0725	0.0698
	SGDE-S	0.0919	0.0895	0.0894	0.0903
	SGDE	0.0900	0.0877	0.0870	0.0880
	RSGDE	0.0947	0.0919	0.0917	0.0924
	CSGDE	**0.0966**	**0.0938**	**0.0933**	**0.0939**
ML-1M	LightGCN	0.5917	0.5261	0.5941	0.5031
	SGDE-S	0.6458	0.5702	0.6466	0.5421
	SGDE	0.6491	0.5730	0.6496	0.5445
	RSGDE	**0.6559**	**0.5771**	**0.6554**	**0.5468**
	CSGDE	**0.6581**	**0.5798**	**0.6583**	**0.5502**
Gowalla	LightGCN	0.1900	0.1677	0.1690	0.1484
	SGDE-S	0.1820	0.1607	0.1628	0.1428
	SGDE	0.1820	0.1607	0.1628	0.1428
	RSGDE	**0.1917**	**0.1690**	**0.1691**	**0.1485**
	CSGDE	**0.1950**	**0.1714**	**0.1712**	**0.1496**

CSGDE: with all designs

RSGDE: with only L_main loss

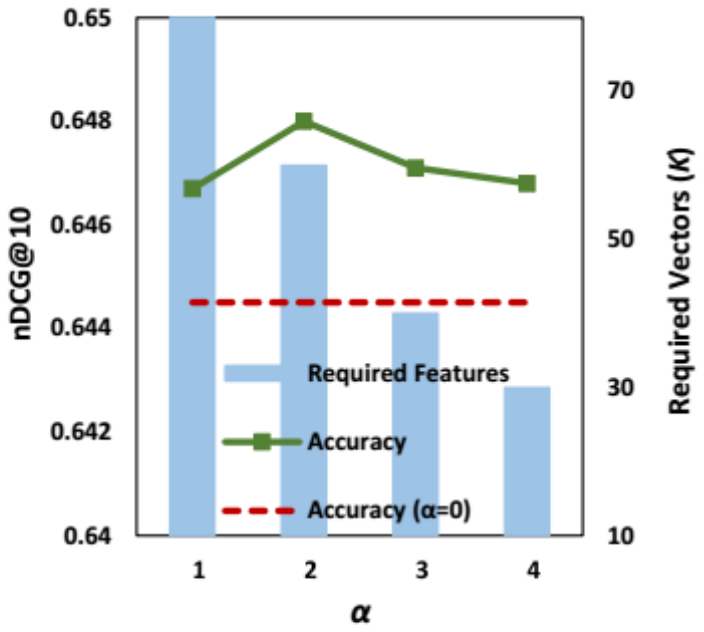
SGDE: RSGDE without adversarial training

SGDE-S: without W, no training required

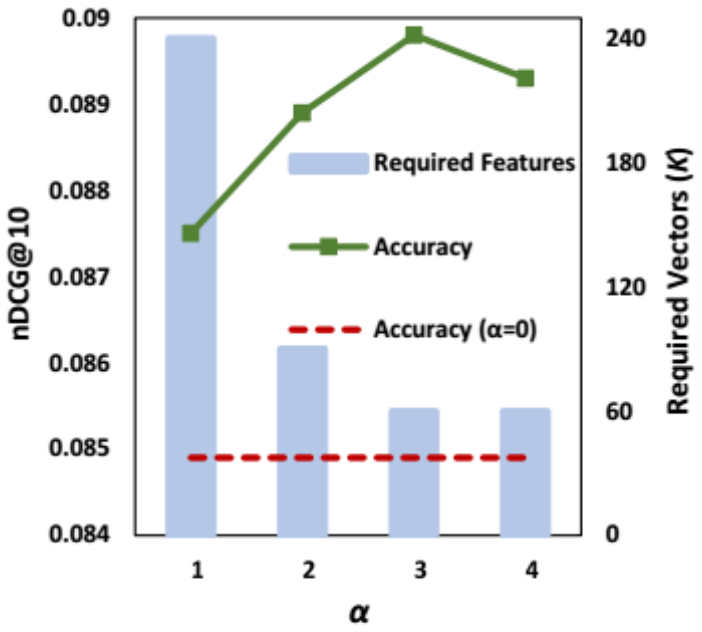
Efficiency

Dataset	Model	Time/Epoch	Epochs	Running Time	Parameters
Yelp	LightGCN	3.66s	180	658.8s	3.3m
	UltraGCN	1.40s	60	84.00s	3.3m
	BPR	0.77s	330	254.10s	3.3m
	GDE	0.97s	150	213.5s	3.3m
	SGDE	0.74s	8	7.660s	4.1k
	RSGDE	0.78s	8	7.980s	4.1k
	CSGDE	1.83s	8	16.38s	4.1k
ML-1M	LightGCN	4.76s	270	1285.2s	0.64m
	UltraGCN	1.50s	25	37.50s	0.64m
	BPR	0.80s	120	96.00s	0.64m
	GDE	1.00s	40	40.30s	0.64m
	SGDE	0.60s	10	6.820s	4.1k
	RSGDE	0.87s	10	9.520s	4.1k
	CSGDE	1.80s	10	18.82s	4.1k
Gowalla	LightGCN	6.43s	600	3,858s	4.5m
	UltraGCN	2.55s	90	229.5s	4.5m
	BPR	1.48s	250	370.0s	4.5m
	GDE	1.95s	120	281.0s	4.5m
	SGDE	1.28s	8	13.31s	5.7k
	RSGDE	1.32s	8	13.63s	5.7k
	CSGDE	3.05s	8	27.47s	5.7k

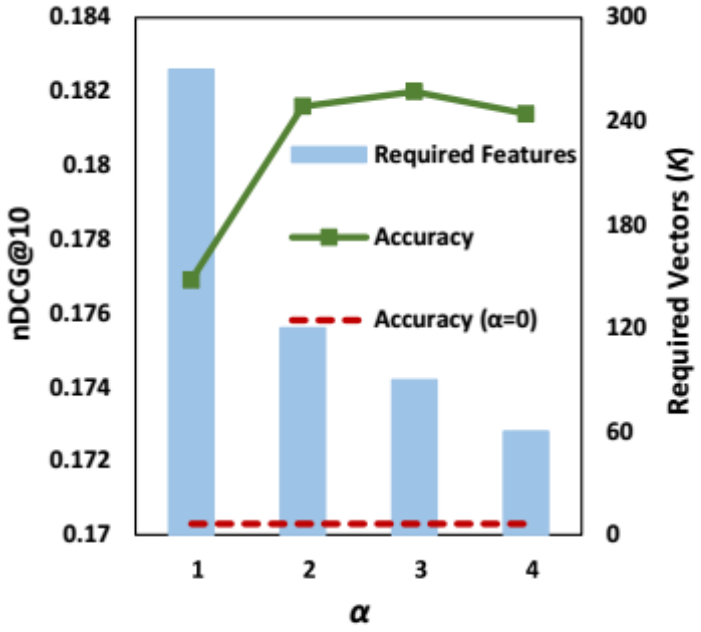
Remove Distribution Redundancy



(a) ML-1M ($K = 120$ when $\alpha = 0$).



(b) Yelp ($K = 2000$ when $\alpha = 0$).



(c) Gowalla ($K = 3000$ when $\alpha = 0$).

Thanks for Your Kind Attention!
ご清聴ありがとうございました。

質疑応答は日本語オッケー