Graph Filtering-Based Collaborative Filtering For Fast Recommendation

Presenter:
Jin-Duk Park

Short Bio.

Jin-Duk Park

Contact: jindeok6@yonsei.ac.kr
Website: jindeok.github.io/jdpark

[Position&Education]

- BS in Yonsei Univ., ME (2012-2018)
- Researcher, LG Electronics (2017 (Intern), 2018-2019 (Full-time))
- **Combined MS/PhD student in Yonsei Univ., CSE (2020~)**
- Visiting researcher: California State Univ., Long Beach (2020), The Univ. of New South Wales, Australia (2023).

[Publication] *(selected)*

- **J.D. Park**, S. Li, X. Cao, and W.Y. Shin, "Criteria Tell you More than Ratings...", In **KDD'23**
- **J.D. Park**, C. Tran, W.Y. Shin, and X. Cao, "On the Power of Gradual Network .....", In **IEEE TPAMI (IF > 24.3)**
- **J.D. Park**, C. Tran, W.Y. Shin, and X. Cao, "Gradual Network...", In **AAAI'22**
- **J.D. Park**, C. Tran, W.Y. Shin, and X. Cao, "GradAlign+: Empowering...", In **CIKM'22**
- K.J. Jeoung, **J.D. Park**, K. Hwang, S.L. Kim, and W.Y. Shin, "Two-Stage Deep Anomaly ...", In **IEEE Access**
- J.C. Moon, Y.M. Shin, **J.D. Park**, N.H. Minaya, W.Y. Shin, and S.I. Choi, "Explainable Gait ....", In **PloSONE**

INDEX

INDEX

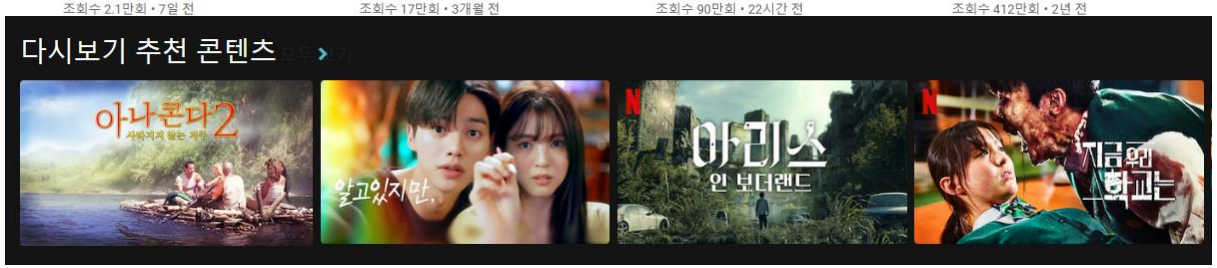
What is Recommender System (RS)?

- RS provides users personalized online product and service recommendations [Jesús et al, 2013]
- Ubiquitous part of today's online entertainment

NAVER

YouTube

NETFLIX



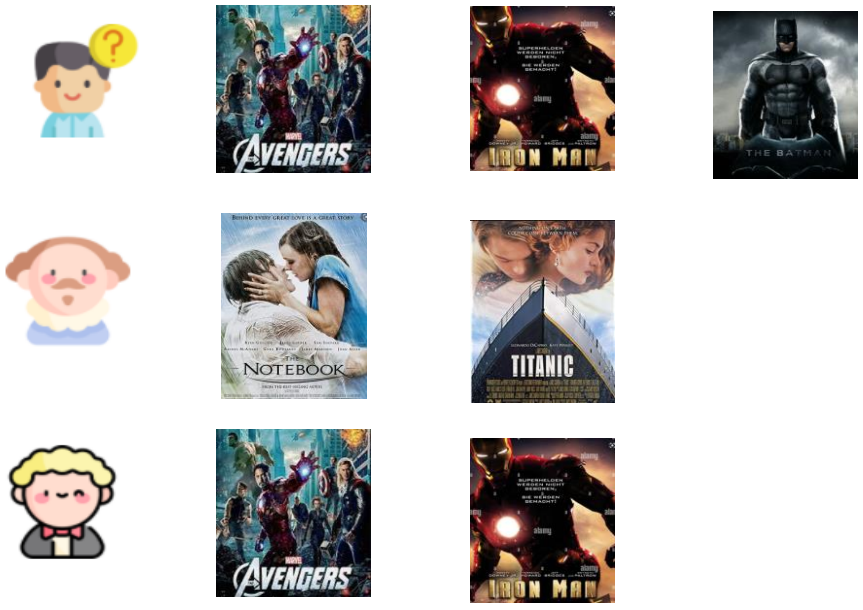
Jesús et al. "Recommender systems survey." Knowledge-based systems 2013

What is Collaborative Filtering (CF)?

- CF aims at finding user/item with **same taste**

  ⟶ Behavioral similarity

- A movie example



What movie would you recommend for  *?*

A Simple CF example

- Starts from **rating matrix** construction
- Problem: what item should be recommended for User 1?

	Item 1	Item 2	Item 3	Item 4
User 1	0	1	1	0
User 2	1	0	0	0
User 3	0	1	1	1

A Simple CF example

- Starts from **rating matrix** construction
- Problem: what item should be recommended for User 1?

	Item 1	Item 2	Item 3	Item 4
User 1	0	1	1	0
User 2	1	0	0	0
User 3	0	1	1	1

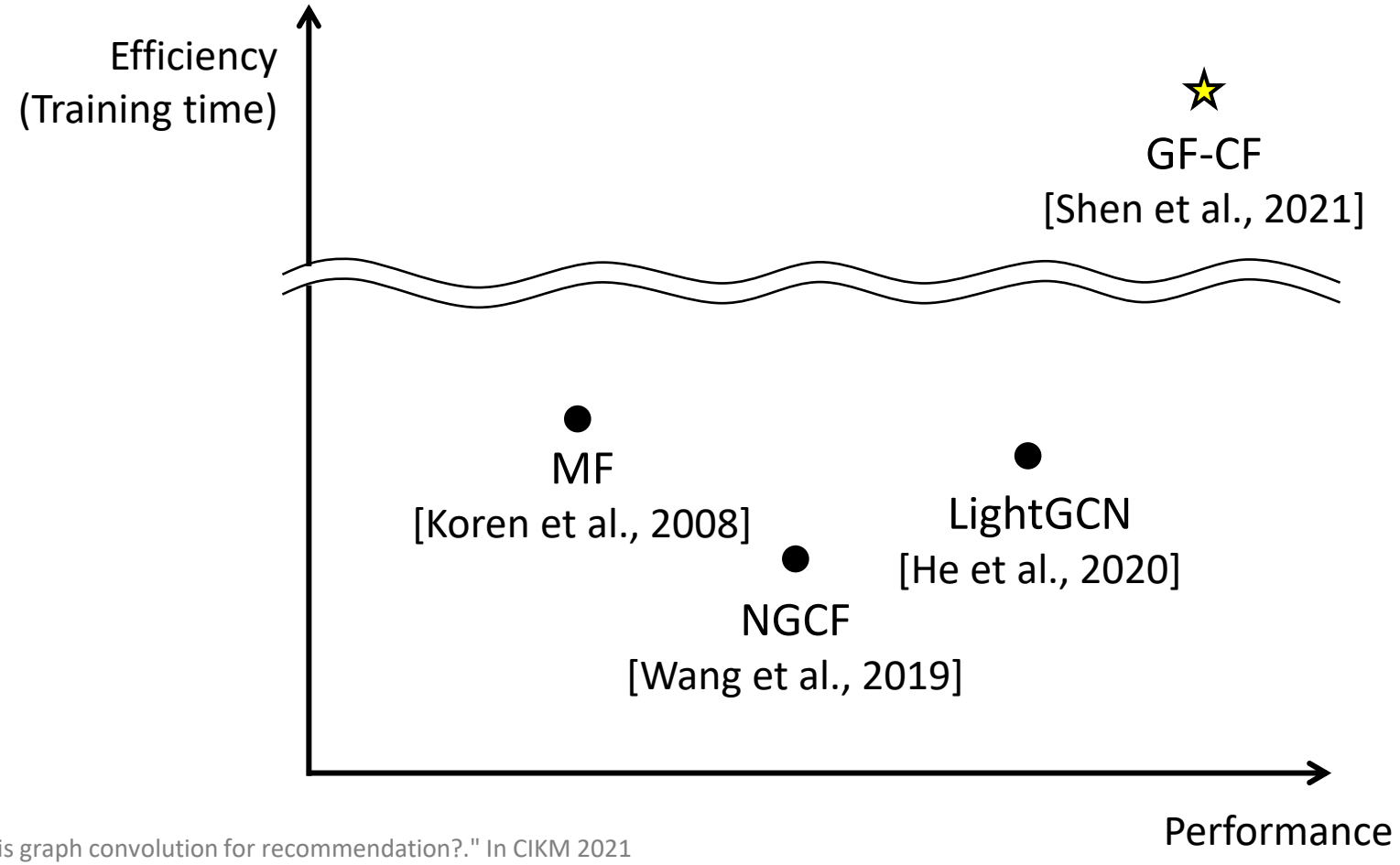
$\boldsymbol{u}_1$=[0,1,1,0] (vector)

$\boldsymbol{u}_2$=[1,0,0,0]

$\boldsymbol{u}_3$=[0,1,1,1]

1. Behavioral similarity: $(\boldsymbol{u}_1, \boldsymbol{u}_3) > (\boldsymbol{u}_1, \boldsymbol{u}_2)$
2. Out of the observed items in $\boldsymbol{u}_3$,
   Recommend **unseen items** for user 1 (item 4)

Graph Filtering-based CF (GF-CF)



Shen et al. "How powerful is graph convolution for recommendation?." In CIKM 2021
He et al. “Lightgcn: Simplifying and powering graph convolution network for recommendation.” In SIGIR 2020
Wang et al. "Neural graph collaborative filtering." In SIGIR 2019
Koren et al. "Modeling relationships at multiple scales to improve accuracy of large recommender systems." In KDD 2007

Graph Filtering-based CF (GF-CF)

GF-CF

[Shen et al., In CIKM 2021]

$$\boldsymbol{s}_u = \boldsymbol{r}_u \left(\tilde{\boldsymbol{R}}^T \tilde{\boldsymbol{R}} + \alpha \boldsymbol{D}_I^{-\frac{1}{2}} \bar{\boldsymbol{U}} \bar{\boldsymbol{U}}^T \boldsymbol{D}_I^{\frac{1}{2}} \right)$$

- Only few lines of code is enough for Implementation
  - **original repo**: https://github.com/yshenaw/GF_CF
  - **faster and simpler version**: https://github.com/jindeok/Linear_GF
- Efficient yet accurate

Shen, Yifei, et al. "How powerful is graph convolution for recommendation?." In CIKM'21

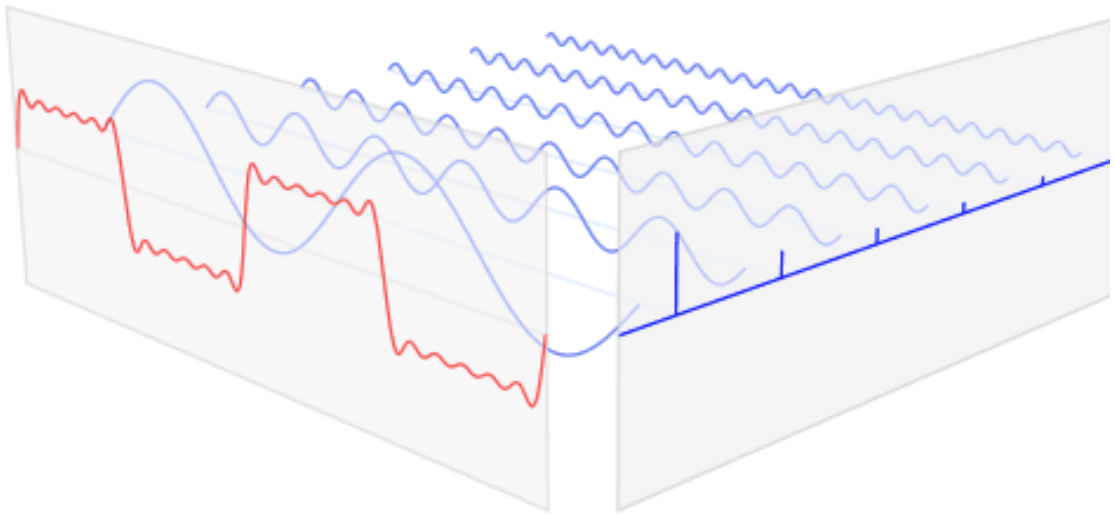
INDEX

Fourier Transform

- **Fourier Transform (FT)**

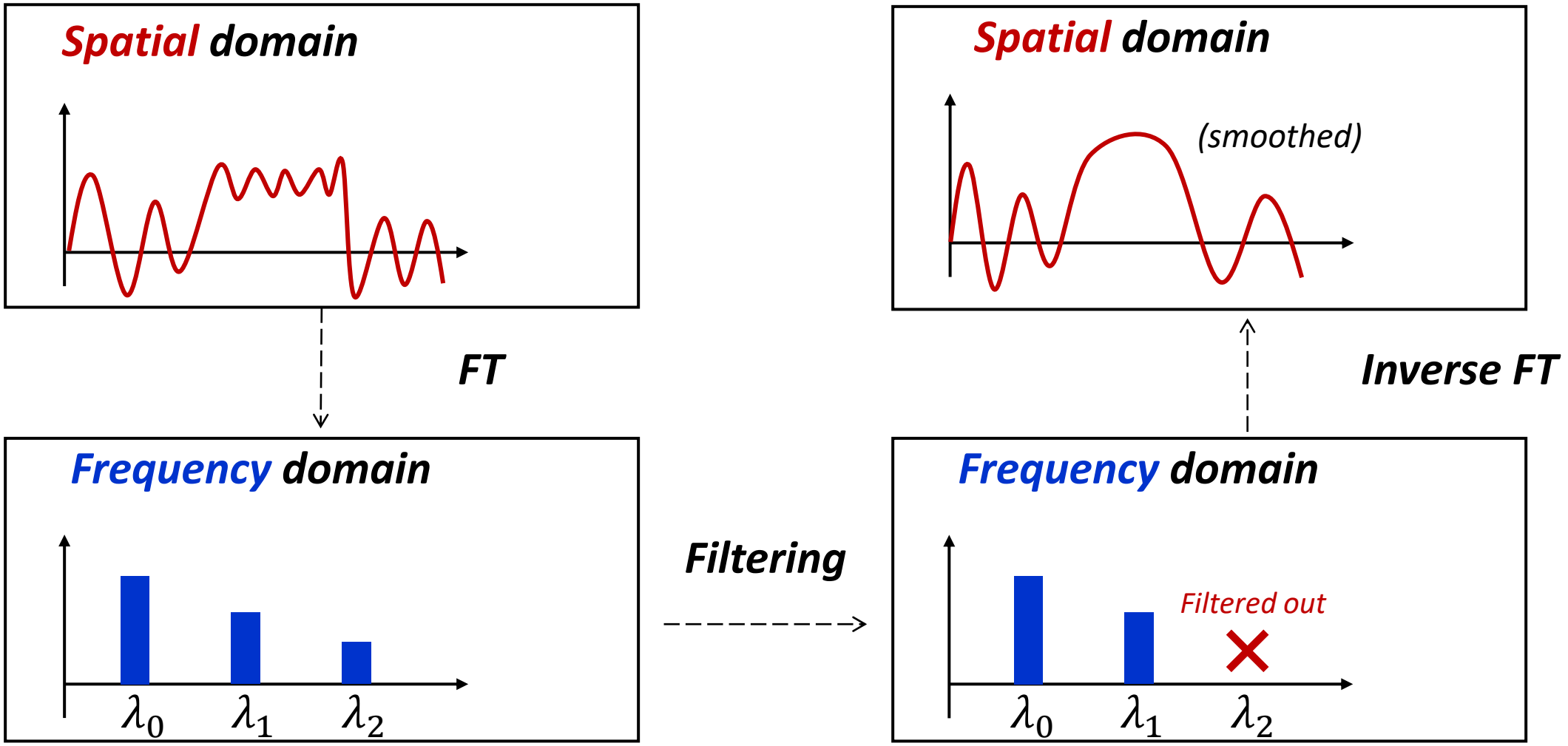
  A mathematical procedure to transform a signal in the time (spatial) domain to a complex number in the frequency domain

$$f(x) \overset{\mathcal{F}}{\longleftrightarrow} \hat{f}(\xi)$$



https://en.wikipedia.org/wiki/Fourier_transform

Overview of Spectral Filtering in Time-Series



Graph Fourier Transform

- **Graph** is a flexible model for representing data in many problems
- **Fourier Transform (FT)**

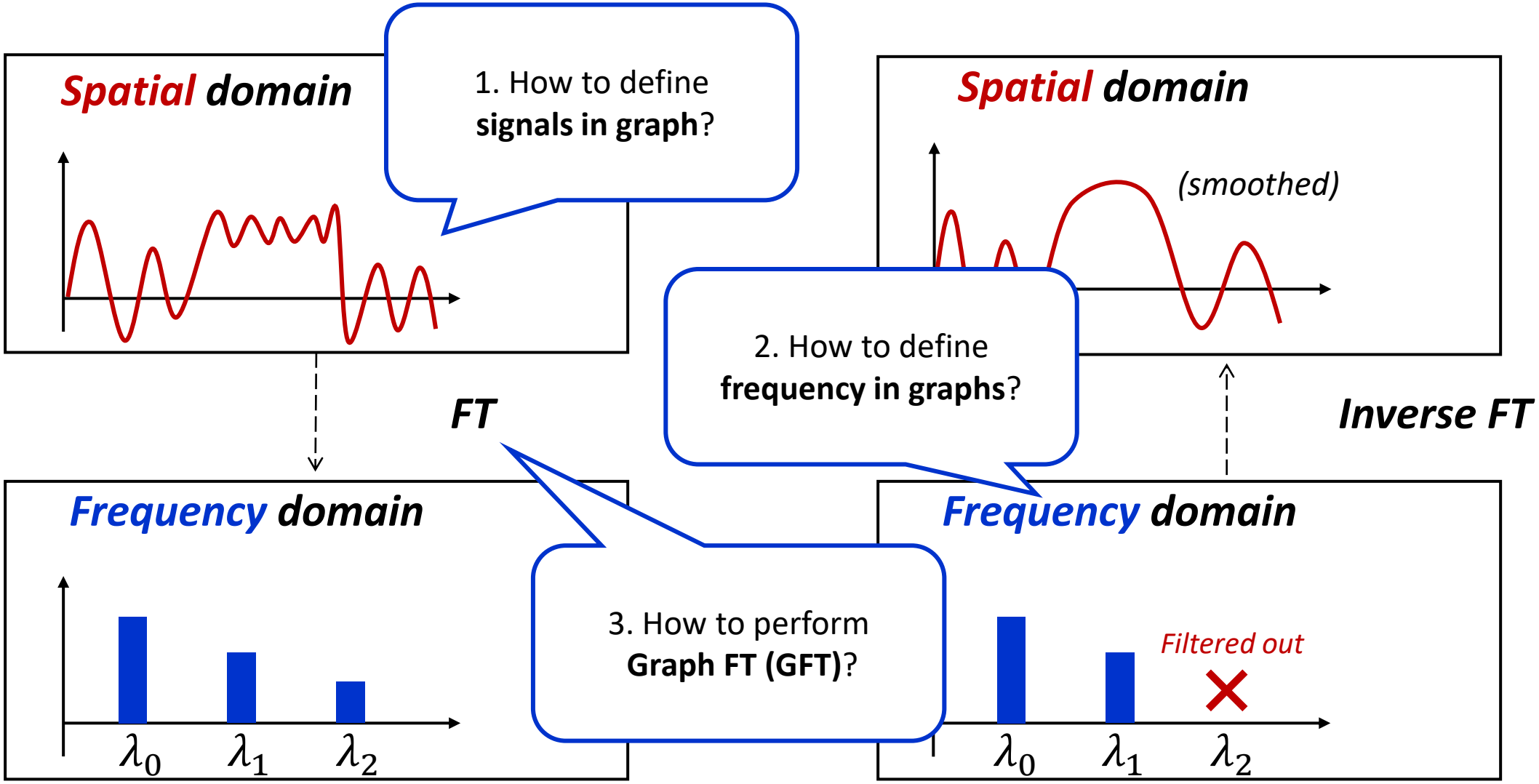
  A mathematical procedure a signal in the time domain to a complex number in the frequency domain

$$f(x) \overset{\mathcal{F}}{\longleftrightarrow} \hat{f}(\xi)$$

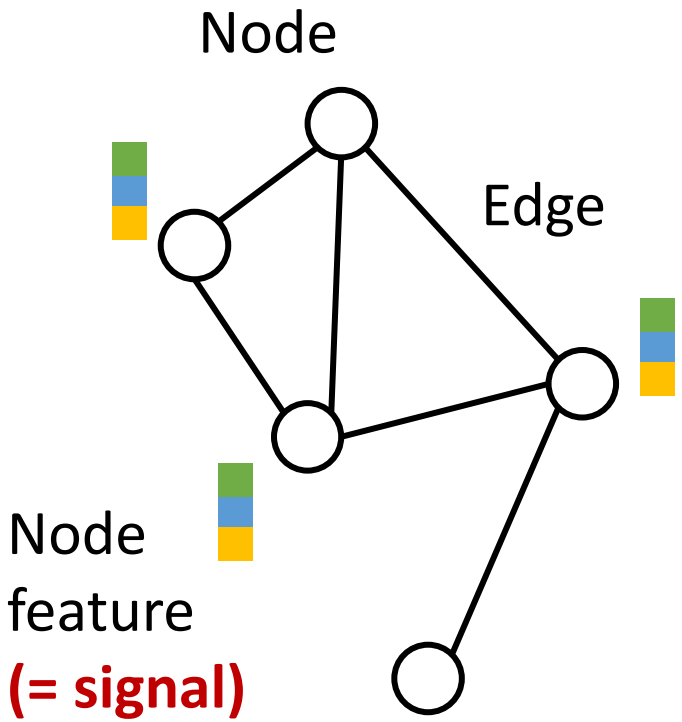
- **Graph Fourier Transform (GFT)**

How to analyze **graph in frequency domain**, via GFT?

Major Challenges in Graph Filtering



Graph and Graph Signals



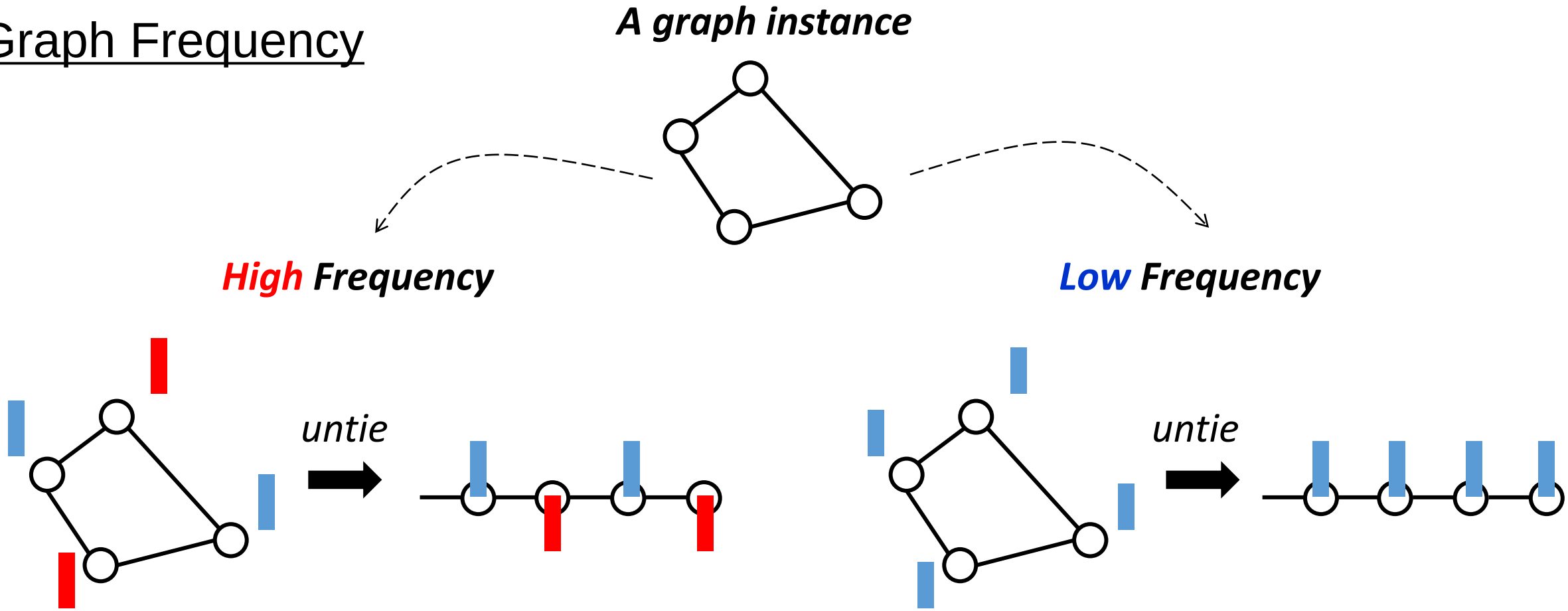
- Nodes (vertices), edges

$$G = (V, E)$$

- **Graph signal**

$$\mathbf{x} = x_1, x_2, \ldots$$

Graph Frequency



***Connected components** have **different** signals*

***Connected components** have **same (similar)** signals*

Graph Frequency and GFT

** All eigenvalues and eigenvectors are sorted with **ascending order***
** We only consider real-valued component of eigenvalues now*

- **Graph Spectral Decomposition**

  - Reference operator should be **diagonalizable**

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \rightarrow \mathbf{R} \in \mathbb{R}^{N \times N} \Rightarrow \mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

$$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \chi_0 & — \\ & \cdots & \\ — & \chi_{N-1} & — \end{bmatrix}$$

Laplacian matrix

- Measure difference between own signal and its neighbors

Reference operator

Eigendecomposition

Definition 1.2 (Graph Fourier Transform). *For a given diagonalizable reference operator* $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ *acting on a graph* $\mathcal{G}$*, the GFT of a graph signal* $\mathbf{x} \in \mathbb{R}^N$ *is:*

$$\mathbb{F}_{\mathcal{G}}\,\mathbf{x} \doteq \hat{\mathbf{x}} \doteq \mathbf{U}^{-1}\mathbf{x}. \tag{2}$$

- **Graph Frequency**

Definition 1.3 (Graph frequency). *Let* $\mathbf{R}$ *be a reference operator. If its eigenvalues are real, the generalized graph frequency* ν *of a graph Fourier mode* $\mathbf{u}_k$ *is:*

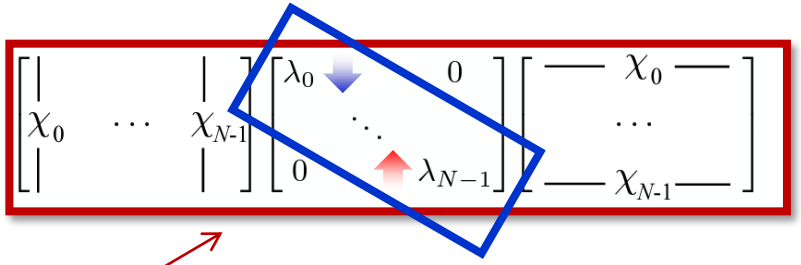
$$\nu(\mathbf{u}_k) = \lambda_k \geqslant 0. \tag{10}$$

$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}$$

Shuman, David I., et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." IEEE signal processing magazine 30.3 (2013): 83-98.

Graph Frequency and GFT

** All eigenvalues and eigenvectors are sorted with **ascending order***
** We only consider real-valued component of eigenvalues now*



- **Graph Spectral Decomposition**

  - Reference operator should be **diagonalizable**

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \rightarrow \mathbf{R} \in \mathbb{R}^{N \times N} \Rightarrow \mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

Laplacian matrix — *Reference operator* — Eigendecomposition

- Measure difference between own signal and its neighbors

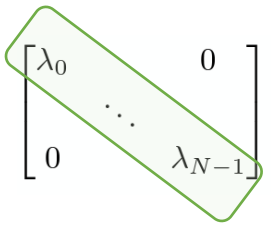
Definition 1.2 (Graph Fourier Transform). *For a given diagonalizable reference operator* $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ *acting on a graph* $\mathcal{G}$*, the GFT of a graph signal* $\mathbf{x} \in \mathbb{R}^N$ *is:*

$$\mathbb{F}_{\mathcal{G}}\,\mathbf{x} \doteq \hat{\mathbf{x}} \doteq \mathbf{U}^{-1}\mathbf{x}. \tag{2}$$

- **Graph Frequency**

Definition 1.3 (Graph frequency). *Let* $\mathbf{R}$ *be a reference operator. If its eigenvalues are real, the generalized graph frequency* ν *of a graph Fourier mode* $\mathbf{u}_k$ *is:*

$$\nu(\mathbf{u}_k) = \lambda_k \geqslant 0. \tag{10}$$



Shuman, David I., et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." IEEE signal processing magazine 30.3 (2013): 83-98.

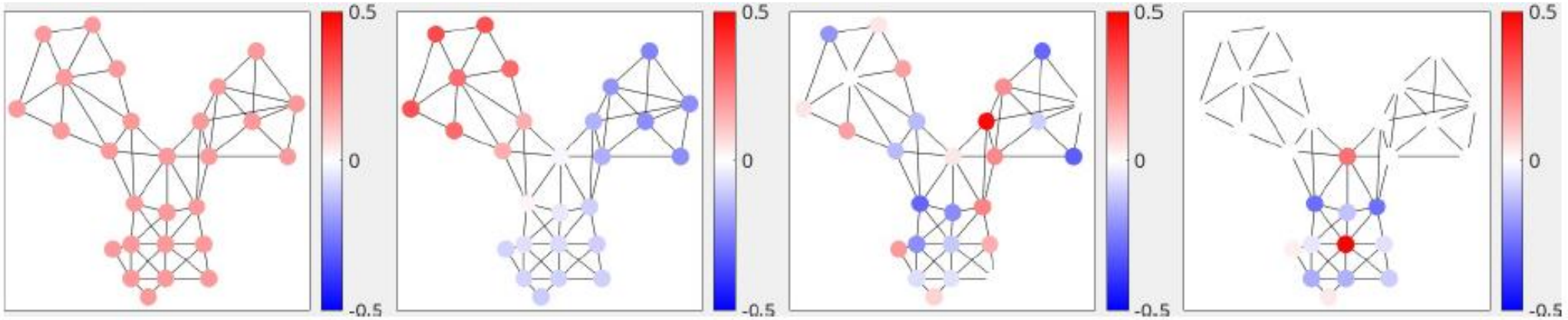
Graph Frequency and GFT

eigenvector

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

$$\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \mathbf{u}_0 & — \\ & \cdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}$$

$\mathbf{u}_0$ $\mathbf{u}_1$ $\mathbf{u}_2$ $\cdots$ $\mathbf{u}_{N-1}$



[Ortega et al., 2018]

Ortega, Antonio, et al. "Graph signal processing: Overview, challenges, and applications." Proceedings of the IEEE 106.5 (2018): 808-828.

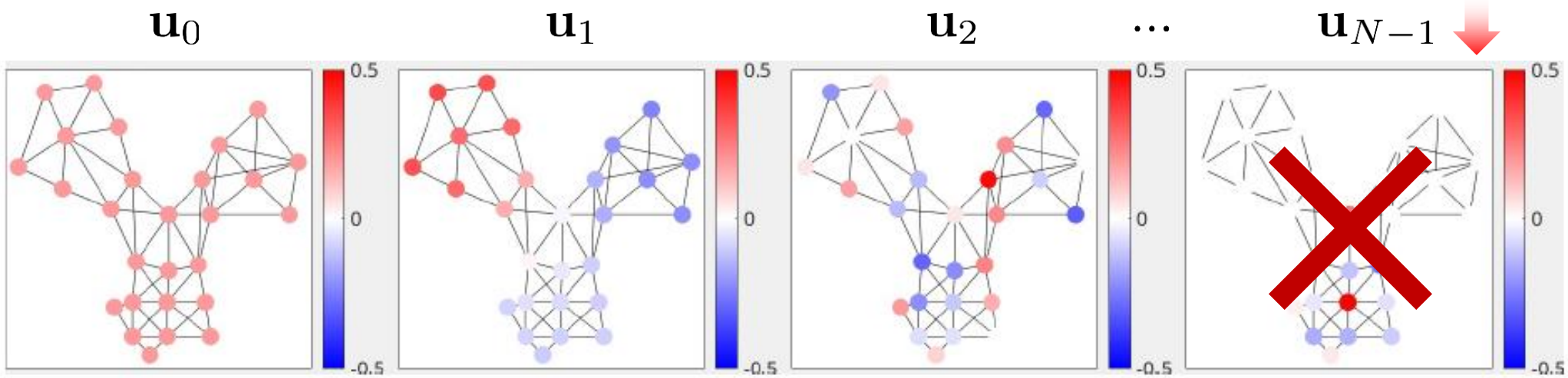
Graph Filtering

** Maps eigenvalues via arbitrary function h*

$$\mathbf{L}^* = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^{-1}$$

$$\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} h(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & h(\lambda_{N-1}) \end{bmatrix} \begin{bmatrix} — & \mathbf{u}_0 & — \\ & \cdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}$$

e.g., **low-pass filter:*
Dampen effect of high frequency part



[Ortega et al., 2018]

Ortega, Antonio, et al. "Graph signal processing: Overview, challenges, and applications." Proceedings of the IEEE 106.5 (2018): 808-828.

What is Graph Convolution?

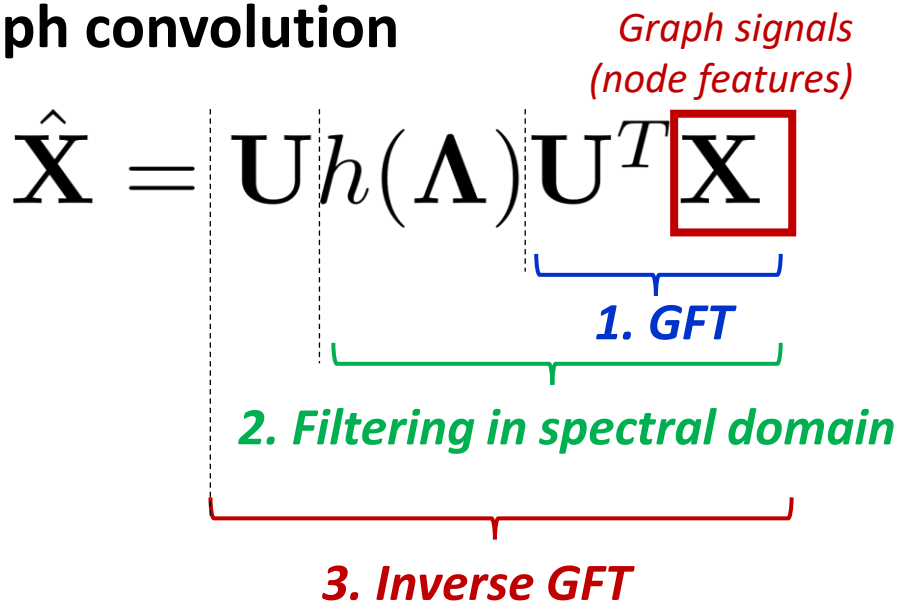
- **Graph convolution**

Graph signals (node features)

$$\hat{\mathbf{X}} = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{X}$$

What is Graph Convolution?

- **Graph convolution**



1. GFT

- Maps to spectral domain

2. Filtering

- Filter out some frequency components

3. Inverse GFT

- Back to original (vertex) domain

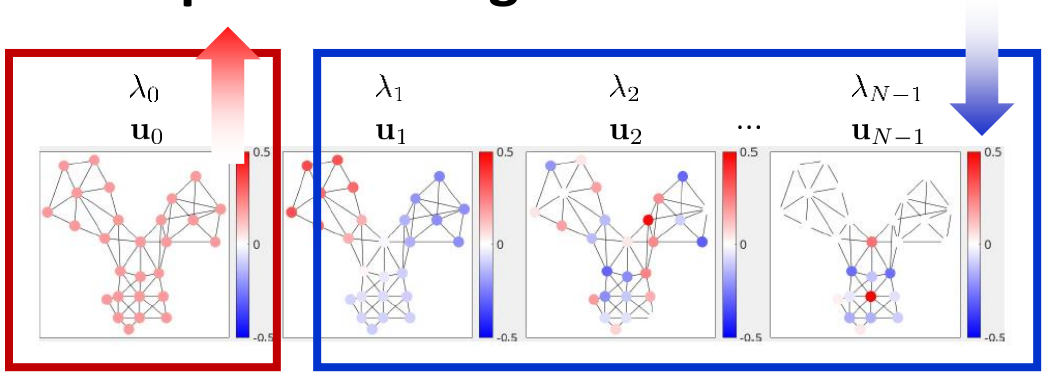
How Graph Convolution Works (Example)?

$$\hat{\mathbf{X}} = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{X}$$

$$\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} h(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & h(\lambda_{N-1}) \end{bmatrix} \begin{bmatrix} — & \mathbf{u}_0 & — \\ & \cdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix} \cdot$$

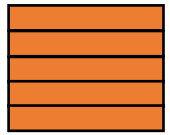
1. Low-pass filtering

(



.

(All-same) Node features

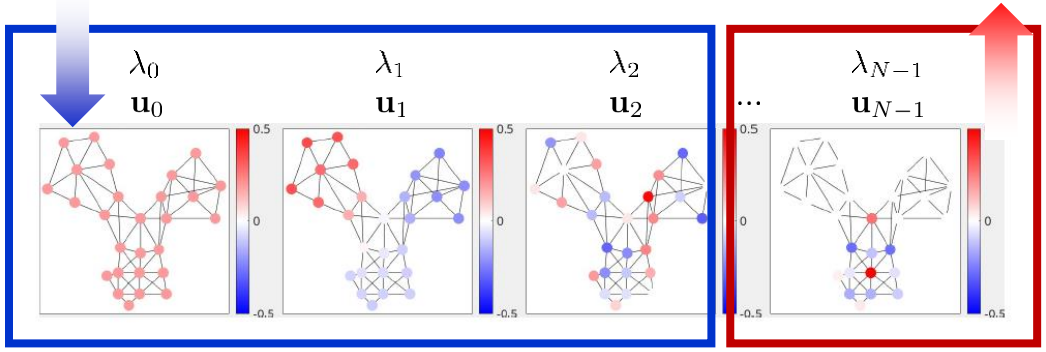


)

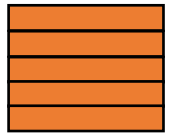
➡ *Higher output value*

2. High-pass filtering

(



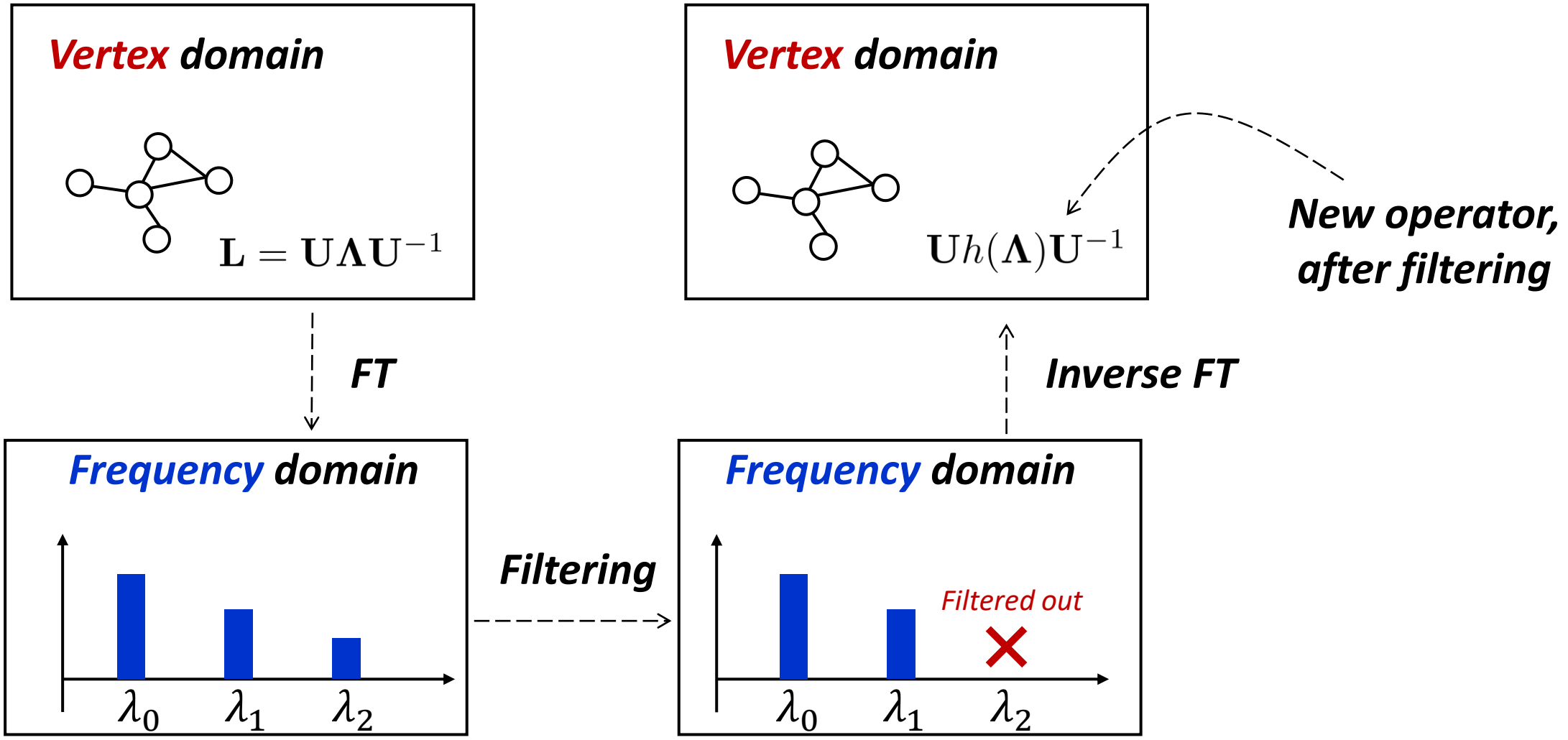
.



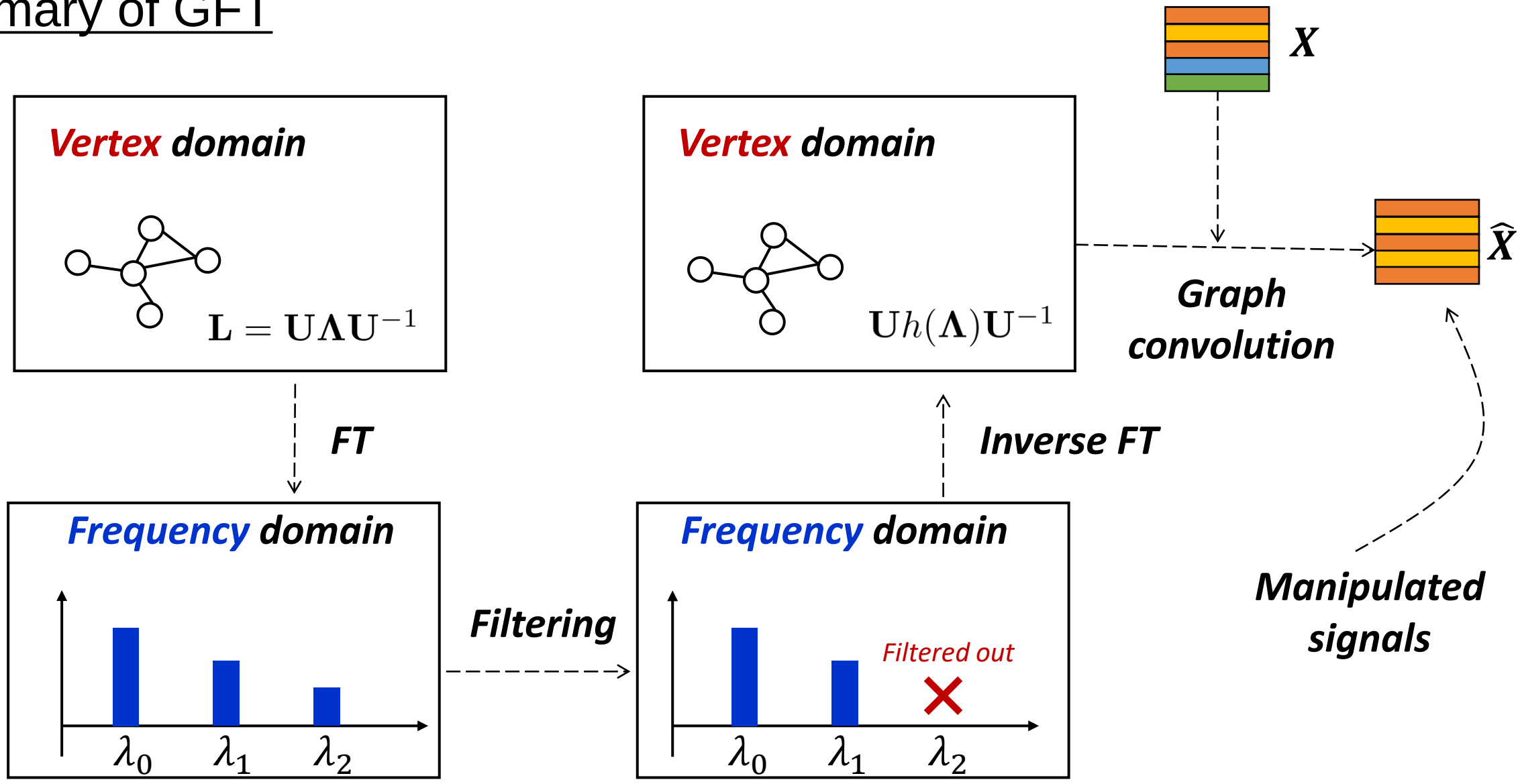
)

➡ *Lower output value (~zero)*

Summary of GFT



Summary of GFT



What is Graph Convolution Network (GCN)?

What if we 'learn' filters?

$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \dashrightarrow \begin{bmatrix} \theta_0 & & 0 \\ & \ddots & \\ 0 & & \theta_{N-1} \end{bmatrix}$$

What is Graph Convolution Network (GCN)?

What if we 'learn' filters?

$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \dashrightarrow \begin{bmatrix} \theta_0 & & 0 \\ & \ddots & \\ 0 & & \theta_{N-1} \end{bmatrix}$$

$$h * x = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^{-1} x$$

Spectral Graph Convolutions

- ***Approximation** by Chebyshev polynomials (Hammond et al., 2011)*

$$h_\theta * x = \sum_{k=0}^{K} \theta_k T_k(\hat{L}) x$$

- ***Linearize with K=1 & stack multiple layers (deeper model)***

$$h_\theta * x = \theta_0 x + \theta_1 (L - I_N) x$$

- ***Integrate two free parameters**, for practicallity*

$$h_\theta * x = \theta (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x$$

- ***Renormalization trick:** Eigenvalues in [0,2] -> stacking layer -> unstable*

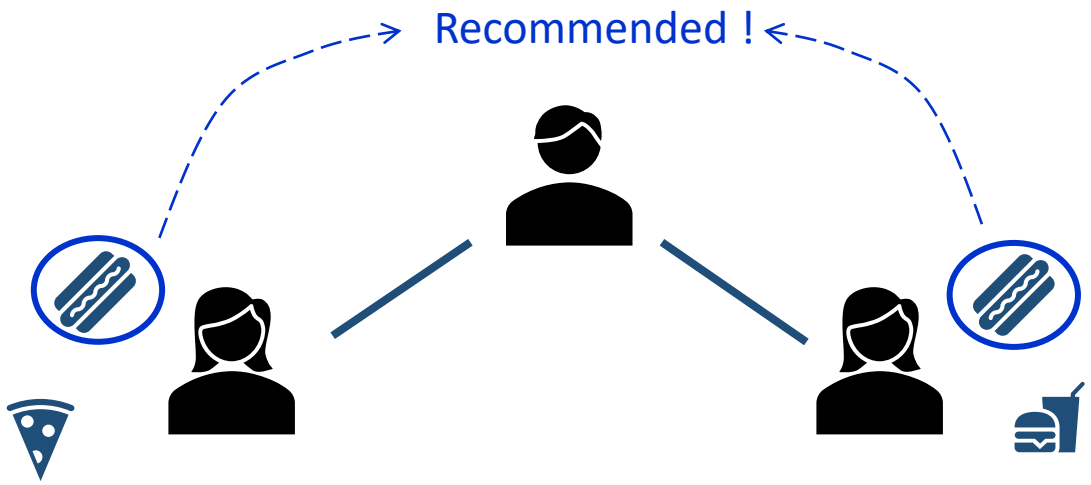
$$Z = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X \Theta$$

Graph Convolutional Networks

INDEX

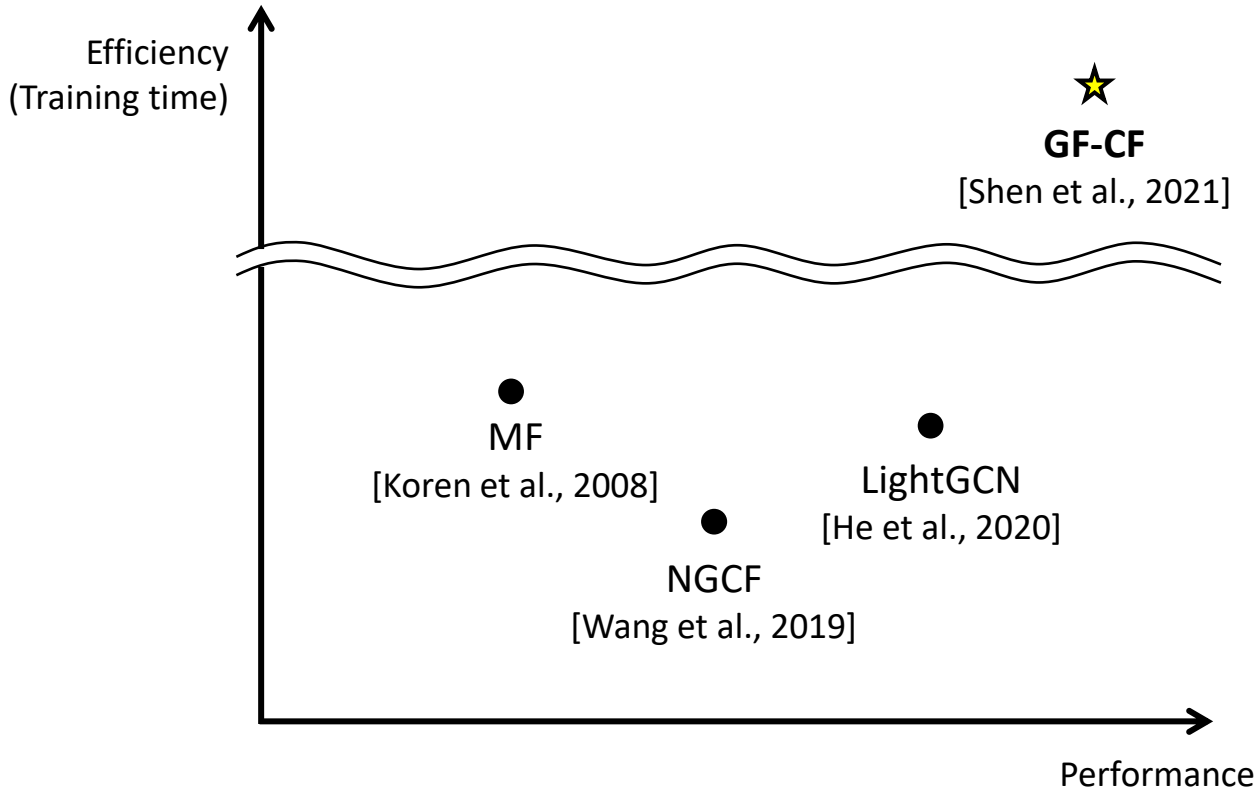
Graph Filtering for Recommendation

- **Low-pass filter** is important for **collaborative filtering (CF)**
  - CF: utilize **similarity** of user/item



GF-CF

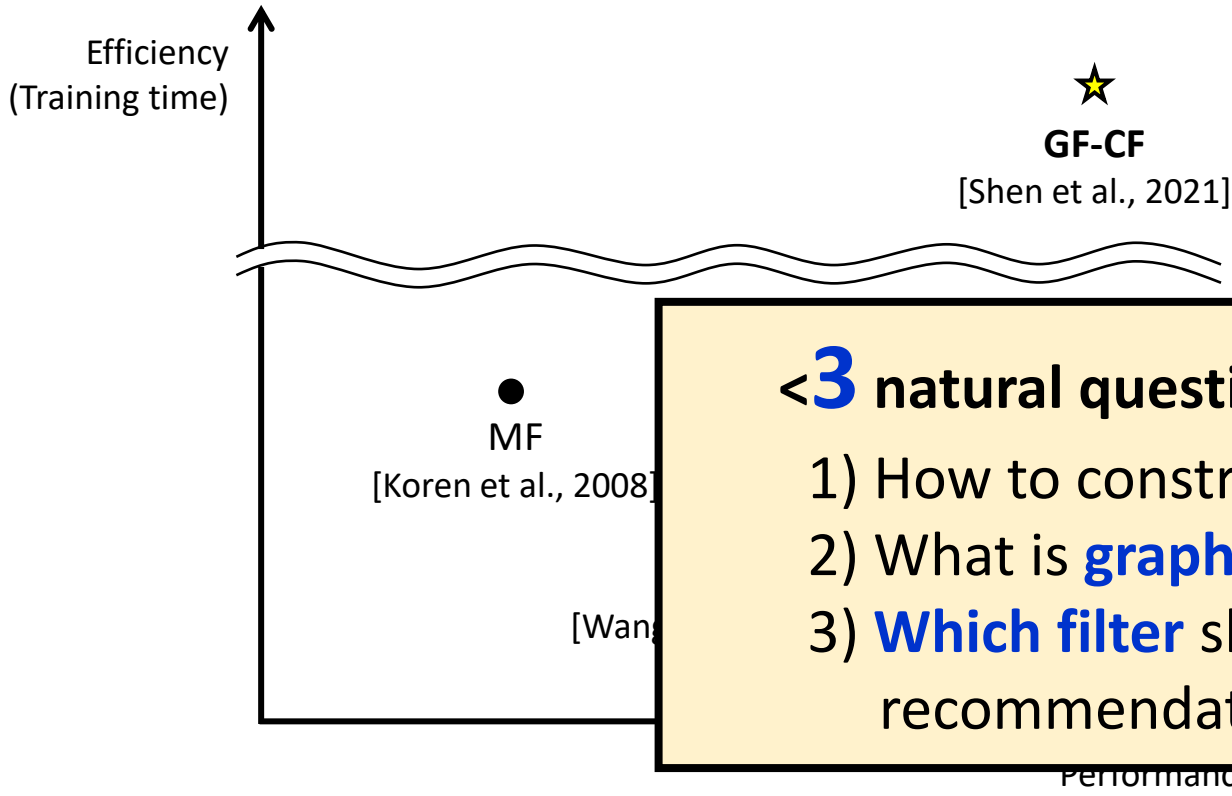
- **GF-CF** [Shen et al., In CIKM 2021] :
  A representative graph filtering method for recommendation



Shen et al. "How powerful is graph convolution for recommendation?." In CIKM 2021

GF-CF

- **GF-CF** [Shen et al., In CIKM 2021] :
  A representative graph filtering method for recommendation



<3 natural questions for graph filtering in CF>

1) How to construct **graph(s)** in CF?
2) What is **graph signals** in CF?
3) **Which filter** should be used for accurate recommendation?

Shen et al. "How powerful is graph convolution for recommendation?." In CIKM 2021

Graph Construction in CF

1) How to construct graph(s) in CF?

2) What is graph signals in CF?

3) Which filter should be used for accurate recommendation?

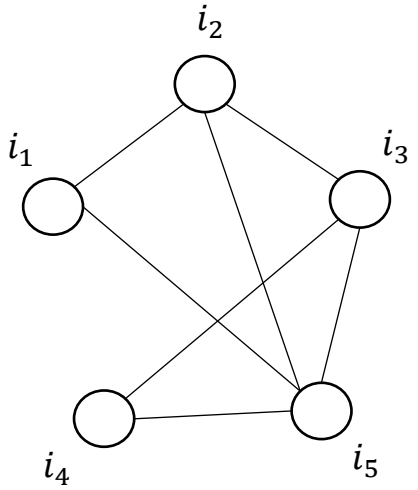
User	Item	Ratings
u_1	i_1	4
u_1	i_2	4
u_2	i_2	4
u_2	i_3	3
	…	

Rating matrix R

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

Normalization: $\tilde{R} = D_U^{-\frac{1}{2}} R D_I^{-\frac{1}{2}}$

Item-item similarity graph



$$\tilde{P} = \tilde{R}^T \tilde{R}$$

Will be used as graph (Adjacency matrix)

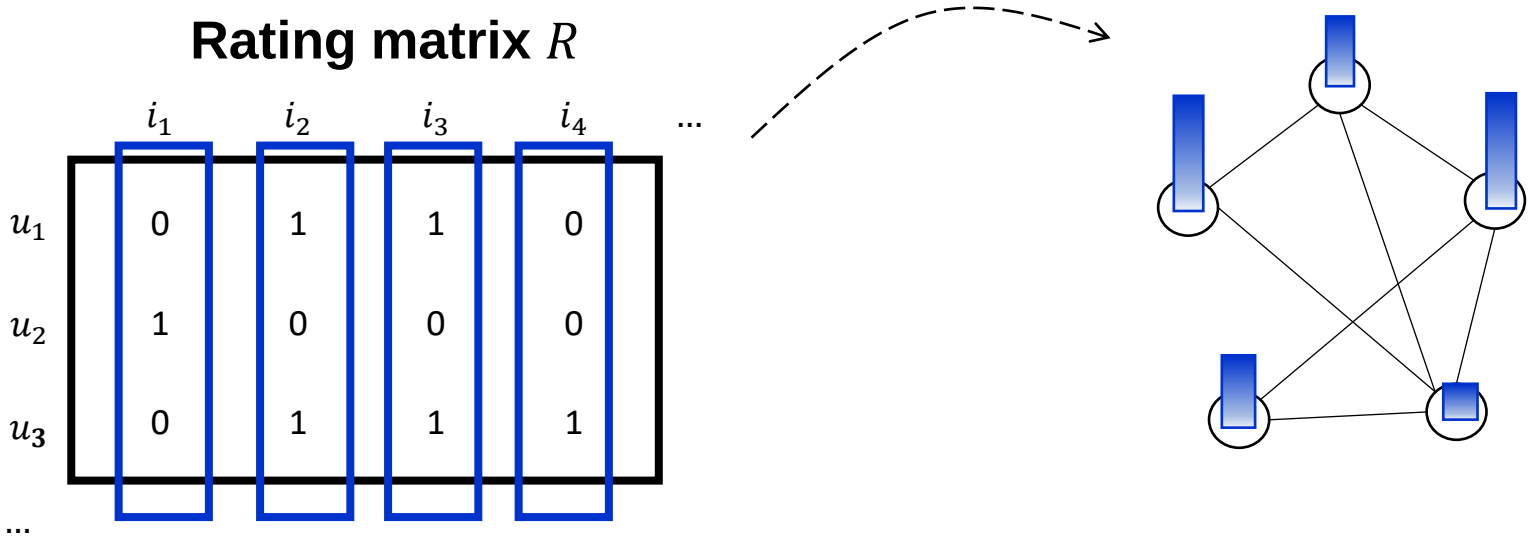
Graph Signals in CF

1) How to construct graph(s) in CF?

2) What is graph signals in CF?

3) Which filter should be used for accurate recommendation?

- We will use R, as graph signals
- Using low-pass filter, promotes low-frequency parts of $\tilde{P}$



GF-CF

1) How to construct graph(s) in CF?
2) What is graph signals in CF?
3) Which filter should be used for accurate recommendation?

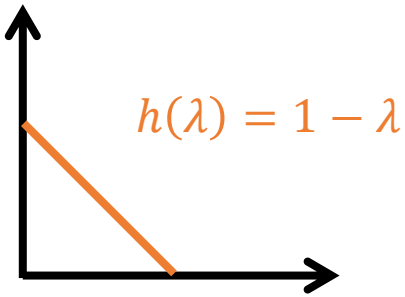
Linear low-pass filter

Ideal low-pass filter

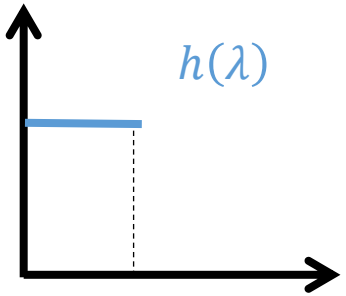
$$s_u = r_u \left(\underbrace{\tilde{R}^T \tilde{R}}_{(=\tilde{P})} + \alpha D_I^{-\frac{1}{2}} \bar{U} \bar{U}^T D_I^{\frac{1}{2}} \right)$$

Low-pass graph filter

- Linear low-pass filter: $1 - \lambda$ (since $\tilde{\mathrm{L}} = I - \tilde{P}$)



- Ideal low-pass filter:



The Results

Dataset	Gowalla			Yelp2018			Amazon-book		
Method	recall	ndcg	training time	recall	ndcg	training time	recall	ndcg	training time
LightGCN-64	0.1830	0.1554	2.77×10^4s	0.0649	0.0530	5.15×10^4s	0.0411	0.0315	1.27×10^5s
LightGCN-128	0.1878	0.1591	3.31×10^4s	0.0671	0.0550	5.66×10^4s	0.0459	0.0353	1.81×10^5s
LightGCN-256	0.1893	0.1606	4.54×10^4s	0.0689	0.0568	8.09×10^4s	0.0481	0.0371	2.98×10^5s
LightGCN-512	0.1892	0.1604	7.28×10^4s	0.0689	0.0569	1.33×10^5s	0.0485	0.0375	5.26×10^5s
GF-CF	0.1849	0.1518	30.5s	0.0697	0.0571	46.0s	0.0710	0.0584	65.8s

- **Accurate performance** w/o expensive training costs
- Much more **faster score calulation** than other neural networks-based models

Shen et al. "How powerful is graph convolution for recommendation?." In CIKM 2021

Further Investigation

Linear low-pass filter

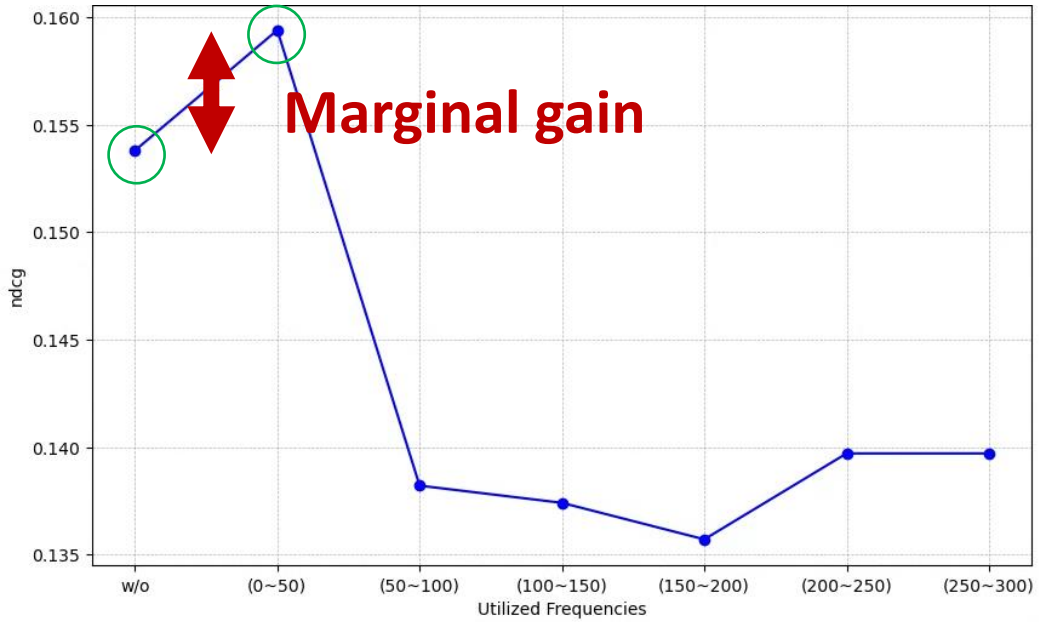
Ideal low-pass filter

$$s_u = r_u \left(\tilde{R}^T \tilde{R} + \alpha D_I^{-\frac{1}{2}} \bar{U} \bar{U}^T D_I^{\frac{1}{2}} \right)$$

: Use **top-k lowest frequency**

- **Ablation study**

  - Dataset: Yahoo! Movie
  - NDCG with **varying k** in the ideal low-pass filter
  - Performance **gain is marginal**!



Further Investigation

Linear low-pass filter

Ideal low-pass filter

$$s_u = r_u \left(\tilde{R}^T \tilde{R} + \alpha D_I^{-\frac{1}{2}} U J^T D_I^{\frac{1}{2}} \right)$$

- **Demerits?** **Ideal low-pass filter** requires **matrix decomposition** (often $\boldsymbol{O(N^3)}$)

- **Do we really need Ideal low-pass filter**?

  - **If not**, we can calculate it **much faster** with GPU acceleration **($S = R\tilde{P}$)**

The Results

- Implemented a **simpler GF method (Linear-GF)**

$$S = R(\tilde{R}^T\tilde{R} + \alpha D_I^{-\frac{1}{2}}\bar{U}\bar{U}^T D_U^{-\frac{1}{2}}) \quad \Rightarrow \quad S = R(\tilde{R}^T\tilde{R})$$

- W/o decomposition, we can now directly accelerate it with GPU!
  ** GPU: NVIDIA RTX A6000

Dataset:Yelp

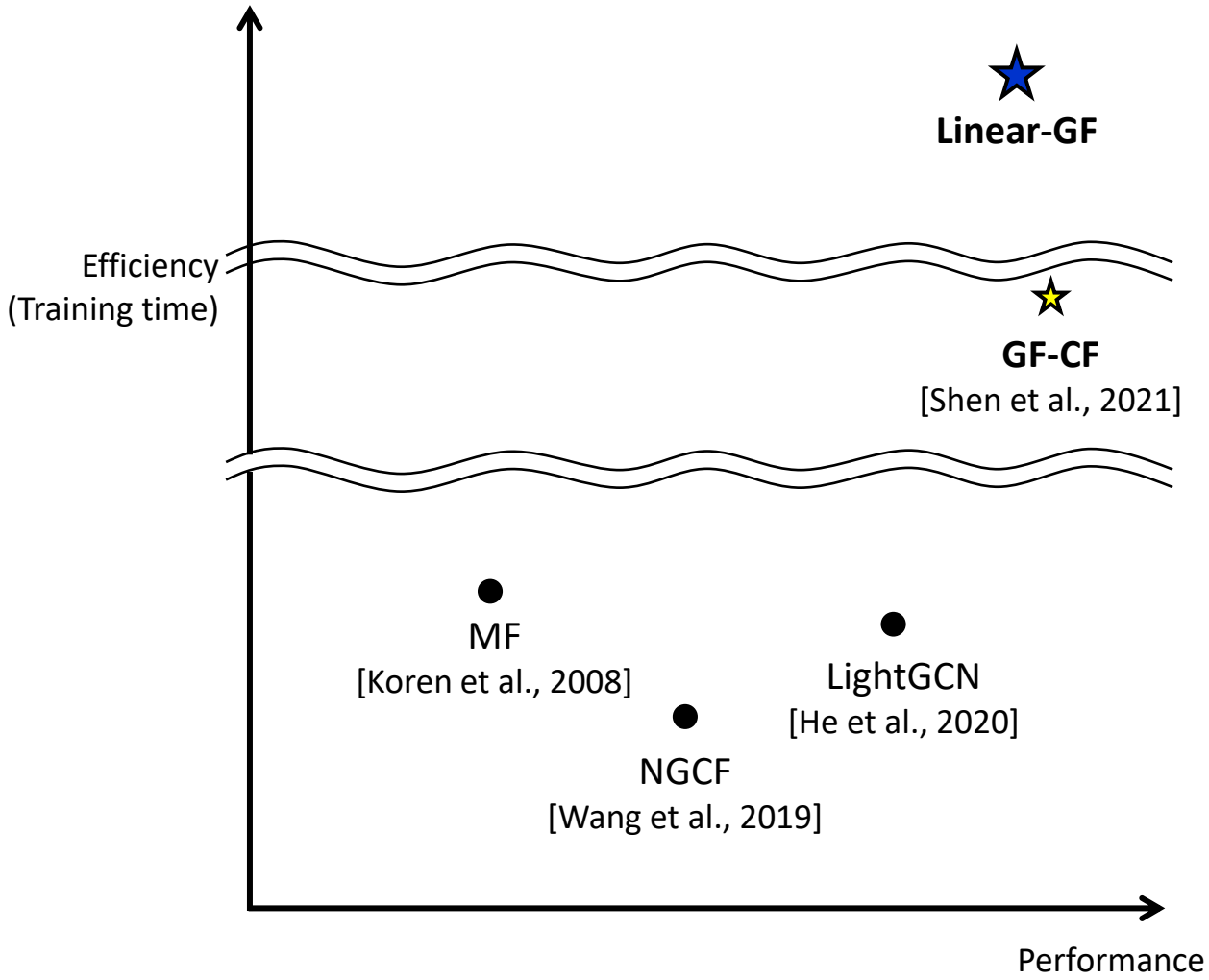
	LightGCN	GF-CF	Linear-GF
Recall@10	0.0530	0.0697	0.0684
Time (sec)	53,482	67.5	0.2

- **Linear-GF achieves extremely faster computation (up to 337x faster) w/o losing much accuracy!**

Linear-GF

- **Linear-GF**: Same as neighborhood-based method

$$S = R(\tilde{R}^T \tilde{R})$$



The Source Code of Linear-GF

```
r_norm = normalize_sparse_adjacency_matrix(R_tr.to_dense(), 0.5)
R = R_tr.to_dense()
P = r_norm.T @ r_norm
P = torch.tensor(P, dtype = torch.float32, device = device)
R = torch.tensor(R, dtype = torch.float32, device = device)

results = R@(P)
```

✓ 0.3s

Refer to the source code of Linear-GF:

https://github.com/jindeok/Linear_GF

- Utilized sparse matrix + gpu acceleration easily with **Scipy** and **PyTorch cuda**
- We supports **4 benchmark** datasets (amazon-book, yelp, gowalla, movielens-1M)
- Refer to the codes for your custom implementation

Takeaways

- What we're aiming to learn in CF domain may lies in simpler space (?)
- **Graph filtering** is a **powerful, training-free method** for accurate recommendation
- However, a **linear low-pass filter** **is still enough** for accurate CF!

$$S = R(\tilde{R}^T \tilde{R})$$

"Success is not final, failure is not fatal:
it is the courage to continue that counts.“
- Winston Churchill

Contact: jindeok6@yonsei.ac.kr
Web: https://jindeok.github.io/jdpark/