EECE 571F: Deep Learning with Structures

Lecture 4: Graph Neural Networks
Graph Convolution Models

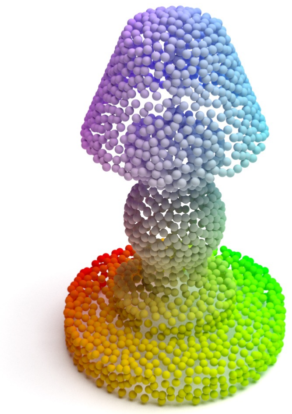
Renjie Liao

University of British Columbia

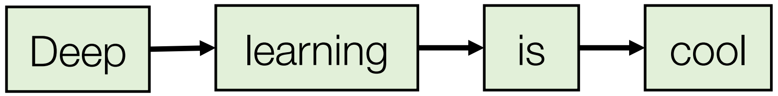
Winter, Term 2, 2021/22

Course Scope

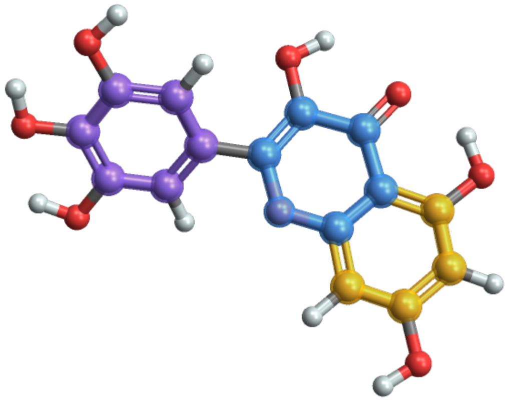
- **Supervised Learning with Observable Structures**
- Unsupervised / Self-supervised Learning with Observable Structures
- Supervised Learning with Latent Structures



Points/Sets



Lists/Sequences



Graphs

Image Credit: https://github.com/AnTao97/PointCloudDatasets
https://www.wolfram.com/language/12/molecular-structure-and-computation/molecule-graphs.html.en?product=mathematica

Deep Learning for Graphs

Graph Neural Networks (GNNs)

- Neural networks that can process general graph structured data
- First proposed in 2008 [1] and dates back to Recursive Neural Networks (mainly processing trees) in 90s [2]
- In fact, Boltzmann Machines [3] (fully connected graphs with binary units) in 80s can be viewed as GNNs
- Most of GNNs (if not all) can be incorporated by the **Message Passing** paradigm
- GNNs have been independently studied in signal processing community under **Graph Signal Processing** [4,5]
- The study of GNNs and other related models are also called **Geometric Deep Learning** [6]

Convolution on Graphs?

- Let us review Fourier Transform and Convolution Theorem

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t)e^{-2\pi i\xi t}dt$$

i.e., expansion in terms of complex exponentials

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials

Laplacian operator is: $\Delta f = \nabla^2 f = \frac{\partial^2}{\partial t^2} f$

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials

Laplacian operator is: $\Delta f = \nabla^2 f = \frac{\partial^2}{\partial t^2} f$

We have $\Delta(e^{-2\pi i \xi t}) = \frac{\partial^2}{\partial t^2} e^{-2\pi i \xi t} = -(2\pi\xi)^2 e^{-2\pi i \xi t}$

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials

Laplacian operator is: $\Delta f = \nabla^2 f = \frac{\partial^2}{\partial t^2} f$

We have $\Delta(e^{-2\pi i \xi t}) = \frac{\partial^2}{\partial t^2} e^{-2\pi i \xi t} = -(2\pi \xi)^2 e^{-2\pi i \xi t}$

$e^{-2\pi i \xi t}$ **is the eigenfunction of Laplacian operator!**

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

Inverse Fourier transform

$$f(t) = \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi t} d\xi$$

i.e., expansion in terms of complex exponentials

Laplacian operator is: $\Delta f = \nabla^2 f = \frac{\partial^2}{\partial t^2} f$

We have $\Delta(e^{-2\pi i \xi t}) = \frac{\partial^2}{\partial t^2} e^{-2\pi i \xi t} = -(2\pi\xi)^2 e^{-2\pi i \xi t}$

$e^{-2\pi i \xi t}$ **is the eigenfunction of Laplacian operator!**

Fourier Transform

Given signal $f(t)$, the classical Fourier transform is:

Inverse Fourier transform

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt \qquad f(t) = \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi t} d\xi$$

i.e., expansion in terms of complex exponentials

Laplacian operator is:

$$\Delta f = \nabla^2 f = \frac{\partial^2}{\partial t^2} f$$

We have

$$\Delta(e^{-2\pi i \xi t}) = \frac{\partial^2}{\partial t^2} e^{-2\pi i \xi t} = -(2\pi\xi)^2 e^{-2\pi i \xi t}$$

$e^{-2\pi i \xi t}$ **is the eigenfunction of Laplacian operator!**

Convolution on Graphs?

- Let us review Fourier Transform and Convolution Theorem

1. Based on the *eigenfunction of Laplacian operator*, we define Fourier transform

2. Based on the convolution theorem, we can define convolution in Fourier domain

Convolution on Graphs?

- Let us review Fourier Transform and Convolution Theorem

  1. Based on the *eigenfunction of Laplacian operator*, we define Fourier transform
  2. Based on the convolution theorem, we can define convolution in Fourier domain

- How can we generalize convolution to graphs?

  1. What is the Laplacian operator on graph?
  2. How can we define convolution in (graph) Fourier domain?

Convolution

Given signal $f(t)$, filter $h(t)$, the convolution is defined as:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t - \tau)d\tau$$

Convolution

Given signal $f(t)$, filter $h(t)$, the convolution is defined as:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

Convolution Theorem tells us that

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi)e^{2\pi i\xi t}d\xi$$

where $\hat{f}(\xi) = \int_{\mathbb{R}} f(t)e^{-2\pi i\xi t}dt$ and $\hat{h}(\xi) = \int_{\mathbb{R}} h(t)e^{-2\pi i\xi t}dt$

Convolution

Given signal $f(t)$, filter $h(t)$, the convolution is defined as:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau$$

How can we generalize them to graphs?

Convolution Theorem tells us that

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi)e^{2\pi i\xi t}d\xi$$

where $\hat{f}(\xi) = \int_{\mathbb{R}} f(t)e^{-2\pi i\xi t}dt$ and $\hat{h}(\xi) = \int_{\mathbb{R}} h(t)e^{-2\pi i\xi t}dt$

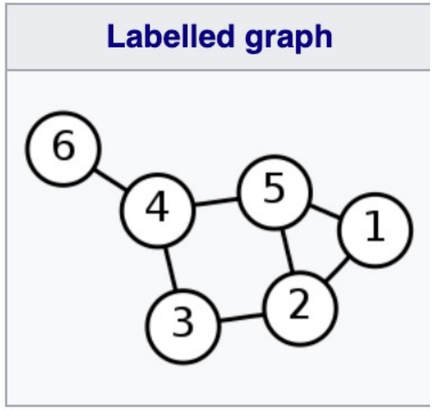
Convolution

How can we generalize convolution to graphs?

Graph Signal

Graph G = (V, E), graph signal (node feature) X

G



A

Adjacency matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Graph Laplacian

Graph G = (V, E), graph signal (node feature) X

Degree matrix: $D_{ii} = \sum_{j=1}^{N} A_{ij}$

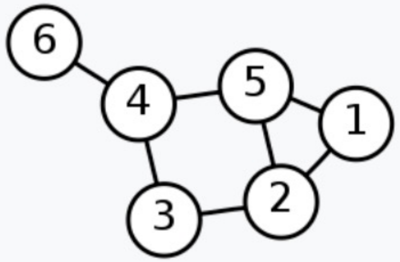
G	D	A
Labelled graph	**Degree matrix**	**Adjacency matrix**
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Graph Laplacian

Graph G = (V, E), graph signal (node feature) X

Degree matrix: $D_{ii} = \sum_{j=1}^{N} A_{ij}$

(Combinatorial) Graph Laplacian: $L = D - A$

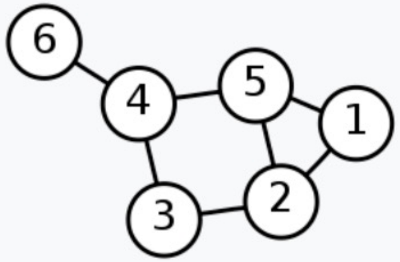
G	D	A	$L = D - A$
Labelled graph	**Degree matrix**	**Adjacency matrix**	**Laplacian matrix**
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Graph Laplacian

Graph G = (V, E), graph signal (node feature) X

Degree matrix: $$D_{ii} = \sum_{j=1}^{N} A_{ij}$$

(Combinatorial) Graph Laplacian: $$L = D - A$$

Compute difference between current node and its neighbors!

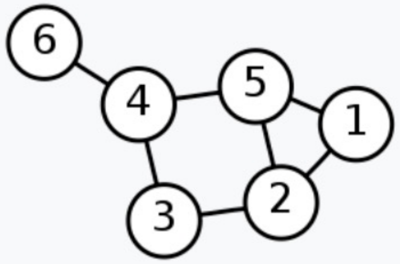
G	D	A	$L = D - A$
Labelled graph	**Degree matrix**	**Adjacency matrix**	**Laplacian matrix**
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Graph Laplacian

For undirected graphs, (Combinatorial) Graph Laplacian:

- Symmetric
- Diagonally dominant
- Positive semi-definite (PSD)
- The number of connected components in the graph the algebraic multiplicity of the 0 eigenvalue.

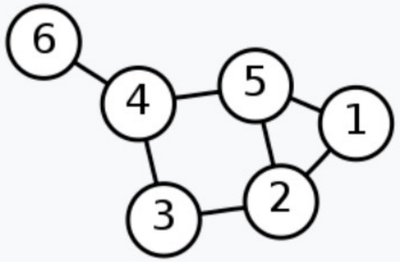
G	D	A	$L = D - A$
Labelled graph	**Degree matrix**	**Adjacency matrix**	**Laplacian matrix**
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Graph Laplacian

Symmetrically Normalized Graph Laplacian:

$$L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

Eigenvalues lie in [0, 2], why? (Try to show it by yourself!)

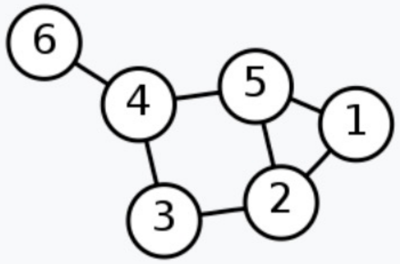
G	D	A	$L = D - A$
Labelled graph	**Degree matrix**	**Adjacency matrix**	**Laplacian matrix**
 	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Image Credit: https://en.wikipedia.org/wiki/Laplacian_matrix

Spectral Theorem

If L is a symmetric matrix, we have

$$L = U \Lambda U^\top = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]$ contains eigenvectors of L and is orthogonal $UU^\top = U^\top U = I$

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix}$$ contains the eigenvalues of L

Spectral Theorem

If L is a symmetric matrix, we have

Spectral Decomposition

$$L = U \Lambda U^\top = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]$ contains eigenvectors of L and is orthogonal $UU^\top = U^\top U = I$

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix}$$

contains the eigenvalues of L

Graph Fourier Transform

Given signal $f(t)$, the *classical Fourier transform* is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials (**eigenfunction of Laplacian operator**)

Graph Fourier Transform

Given signal $f(t)$, the *classical Fourier transform* is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials (**eigenfunction of Laplacian operator**)

Given graph signal $X \in \mathbb{R}^{N \times 1}$, the *Graph Fourier Transform* is:

$$\hat{X}[i] = \sum_{j=1}^{N} U[j, i] X[j]$$

$$\hat{X} = U^{\top} X$$

i.e., expansion in terms of **eigenvectors of Graph Laplacian operator**

Graph Fourier Transform

Given signal $f(t)$, the *classical Fourier transform* is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials (**eigenfunction of Laplacian operator**)

Given graph signal $X \in \mathbb{R}^{N \times 1}$, the *Graph Fourier Transform* is:

$$\hat{X}[i] = \sum_{j=1}^{N} U[j,i] X[j]$$

$$\hat{X} = U^{\top} X$$

Inverse Graph Fourier Transform

$$X = U \hat{X}$$

i.e., expansion in terms of **eigenvectors of Graph Laplacian operator**

Graph Fourier Transform

Given signal $f(t)$, the *classical Fourier transform* is:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

i.e., expansion in terms of complex exponentials (**eigenfunction of Laplacian operator**)

Given graph signal $X \in \mathbb{R}^{N \times 1}$, the *Graph Fourier Transform* is:

$$\hat{X}[i] = \sum_{j=1}^{N} U[j, i] X[j]$$

$$\hat{X} = U^{\top} X$$

Inverse Graph Fourier Transform

Eigenvalue corresponds to frequency!

$$X = U \hat{X}$$

i.e., expansion in terms of **eigenvectors of Graph Laplacian operator**

Graph Convolution (Spectral Filtering)

Convolution:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi)e^{2\pi i\xi t}d\xi$$

Graph Convolution (Spectral Filtering)

Convolution:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi)e^{2\pi i\xi t}d\xi$$

Graph Fourier Transform:

$$\hat{X} = U^\top X \qquad\qquad L = U\Lambda U^\top$$

Graph Convolution (Spectral Filtering)

Convolution:

$$(f * h)(t) = \int_{\mathbb{R}} f(\tau)h(t-\tau)d\tau = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi)e^{2\pi i \xi t}d\xi$$

Graph Fourier Transform:

$$\hat{X} = U^\top X \qquad\qquad L = U\Lambda U^\top$$

Graph Convolution in Fourier domain (Spectral Filtering):

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Spectral Filters

Graph Convolution in Fourier domain (Spectral Filtering):

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Directly construct h requires spectral decomposition which is O(N^3)!

Spectral Filters

Graph Convolution in Fourier domain (Spectral Filtering):

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Directly construct h requires spectral decomposition which is O(N^3)!

Can we find some efficient construction of h?

Spectral Filters

Graph Convolution in Fourier domain (Spectral Filtering):

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Directly construct h requires spectral decomposition which is O(N^3)!

Can we find some efficient construction of h?

- Chebyshev polynomials [7]
- Graph wavelets [7]

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Image Credit: https://en.wikipedia.org/wiki/Chebyshev_polynomials

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

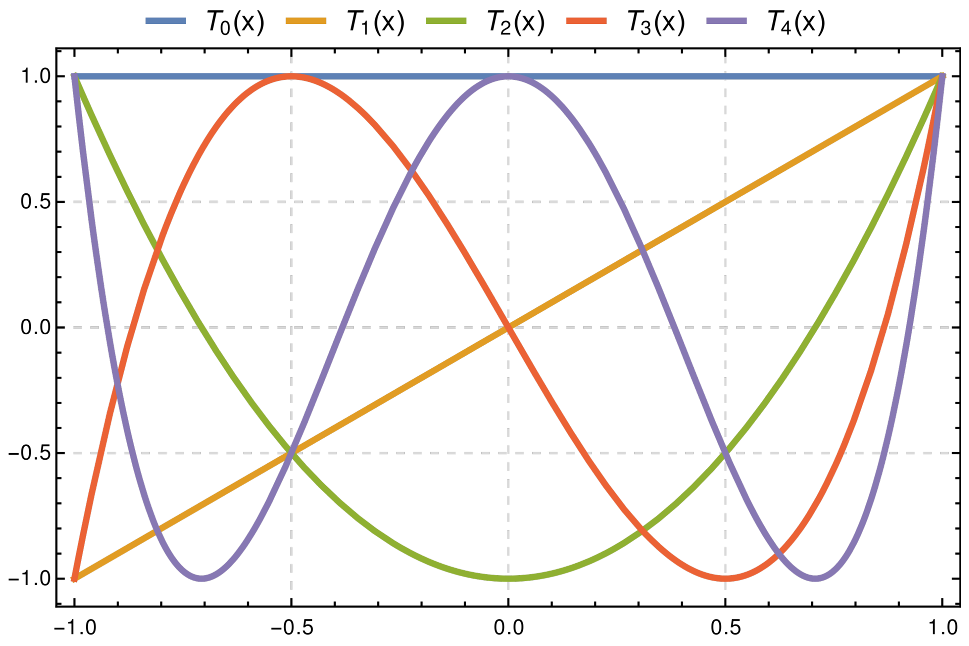
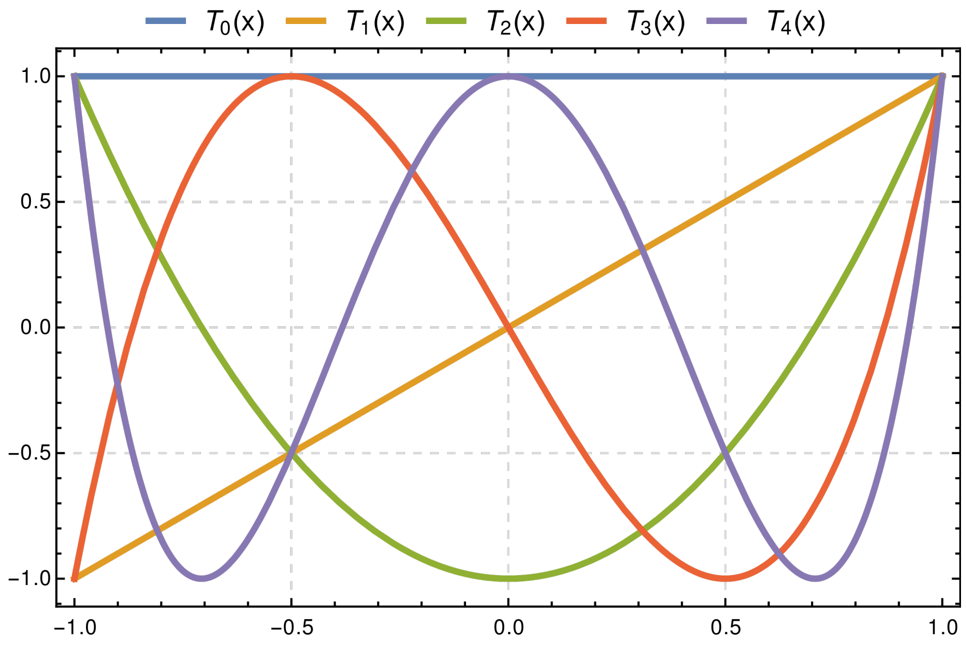


Image Credit: https://en.wikipedia.org/wiki/Chebyshev_polynomials

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$



They provide orthonormal basis in some Sobolev space on [-1, 1]:

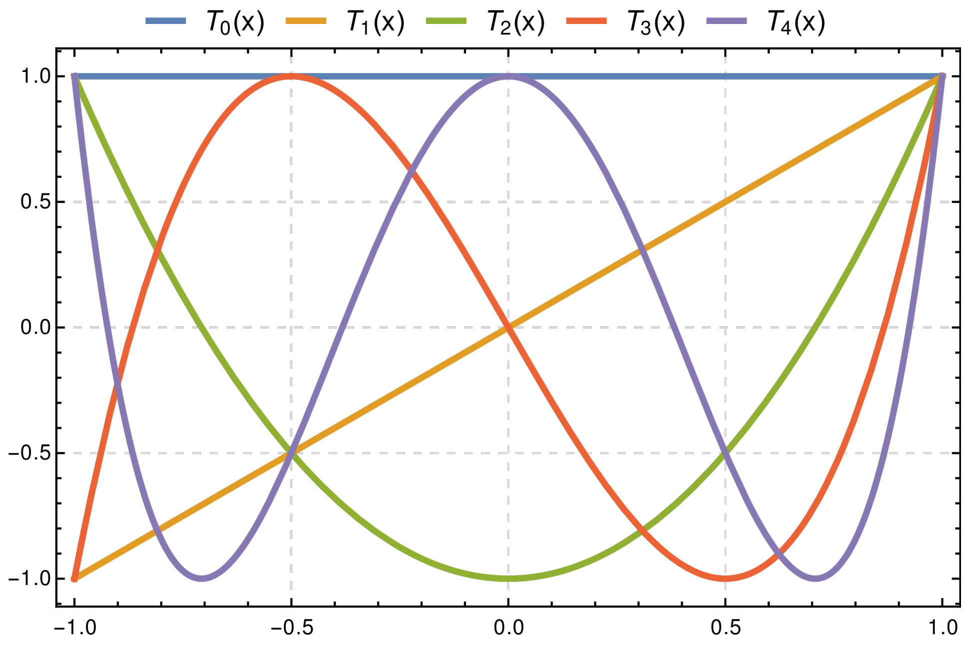
$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

Image Credit: https://en.wikipedia.org/wiki/Chebyshev_polynomials

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$



They provide orthonormal basis in some Sobolev space on [-1, 1]:

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

$$\int_{-1}^{1} T_n(x)T_m(x)\frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0 & \text{if } n \neq m \\ \pi & \text{if } n = m = 0 \\ \frac{\pi}{2} & \text{if } n = m \neq 0 \end{cases}$$

Image Credit: https://en.wikipedia.org/wiki/Chebyshev_polynomials

Spectral Filters

Chebyshev expansion:

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

Spectral Filters

Chebyshev expansion:

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

Spectral filtering:

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Spectral Filters

Chebyshev expansion:

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

Spectral filtering:

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Truncated Chebyshev polynomials approximation:

$$h_\theta(\Lambda) \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{\Lambda}) = \sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I)$$

Spectral Filters

Truncated Chebyshev polynomials approximation:

$$h_\theta(\Lambda) \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{\Lambda}) = \sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I)$$

Graph Convolution:

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Spectral Filters

Truncated Chebyshev polynomials approximation:

$$h_\theta(\Lambda) \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{\Lambda}) = \sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I)$$

Graph Convolution:

$$h_\theta * X = U h_\theta(\Lambda) U^\top X$$

Truncated Chebyshev polynomials based Graph Convolution:

$$\begin{aligned} h_\theta * X &= U h_\theta(\Lambda) U^\top X \\ &\approx U \left(\sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I) \right) U^\top X \end{aligned}$$

Spectral Filters

Recall we do not want explicit spectral decomposition since it is expensive!

$$h_\theta * X \approx U \left(\sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I) \right) U^\top X$$

Spectral Filters

Recall we do not want explicit spectral decomposition since it is expensive!

$$h_\theta * X \approx U \left(\sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I) \right) U^\top X$$

Are Chebyshev polynomials efficient?

Spectral Filters

Recall

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Spectral Filters

Recall

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Let

$$T_n(\tilde{L}) = UT_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top$$

Spectral Filters

Recall

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Let

$$T_n(\tilde{L}) = UT_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top$$

We have

$$\begin{aligned}
T_0(\tilde{L}) &= I \\
T_1(\tilde{L}) &= U\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top = 2L/\lambda_{\max} - I \\
T_{n+1}(\tilde{L}) &= U\left(2\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)T_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right) - T_{n-1}\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)\right)U^\top \\
&= 2U\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top UT_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top - UT_{n-1}\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top \\
&= 2\left(\frac{2L}{\lambda_{\max}} - I\right)T_n(\tilde{L}) - T_{n-1}(\tilde{L})
\end{aligned}$$

Spectral Filters

Recall

$$T_n(\tilde{L}) = U T_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right) U^\top$$

Spectral Filters

Recall

$$T_n(\tilde{L}) = U T_n \left(\frac{2\Lambda}{\lambda_{\max}} - I \right) U^\top$$

We have

$$h_\theta * X \approx U \left(\sum_{n=0}^{K} \theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I) \right) U^\top X$$
$$= \sum_{n=0}^{K} \theta_n T_n(\tilde{L}) X$$

Spectral Filters

Recall

$$T_n(\tilde{L}) = UT_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top$$

We have

$$h_\theta * X \approx U\left(\sum_{n=0}^{K}\theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I)\right)U^\top X$$

$$= \sum_{n=0}^{K}\theta_n T_n(\tilde{L})X$$

Let

$$T_0(\tilde{X}) = T_0(\tilde{L})X$$

Spectral Filters

Recall

$$T_n(\tilde{L}) = UT_n\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^\top$$

We have

$$h_\theta * X \approx U\left(\sum_{n=0}^{K}\theta_n T_n(\frac{2\Lambda}{\lambda_{\max}} - I)\right)U^\top X$$

$$= \sum_{n=0}^{K}\theta_n T_n(\tilde{L})X$$

Let

$$T_0(\tilde{X}) = T_0(\tilde{L})X$$

We have

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right)T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

Spectral Filters

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

where

$$T_0(\tilde{X}) = X$$
$$T_1(\tilde{X}) = 2LX/\lambda_{\text{max}} - X$$
$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\text{max}}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

Spectral Filters

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

where

$$T_0(\tilde{X}) = X$$
$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$
$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

What if we truncate to 1st order?

Spectral Filters

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

where

$$T_0(\tilde{X}) = X$$
$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$
$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

What if we truncate to 1st order?

That is Graph Convolutional Networks (GCNs) [8] !

Graph Convolutional Networks (GCNs)

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

Graph Convolutional Networks (GCNs)

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

Graph Convolutional Networks (GCNs)

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

We can use the normalize graph Laplacian so that its eigenvalues are in [0, 2]

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Graph Convolutional Networks (GCNs)

Truncated Chebyshev polynomials based Graph Convolution:

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

We can use the normalize graph Laplacian so that its eigenvalues are in [0, 2]

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Assuming $\lambda_{\max} \approx 2$

$$\begin{aligned} h_\theta * X &\approx \theta_0 X + \theta_1 T_1(\tilde{X}) \\ &\approx \theta_0 X - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \end{aligned}$$

Graph Convolutional Networks (GCNs)

Simplified Truncated Chebyshev polynomials based Graph Convolution:

$$
\begin{aligned}
h_\theta * X &\approx \theta_0 X + \theta_1 T_1(\tilde{X}) \\
&\approx \theta_0 X - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \\
&= \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) X
\end{aligned}
$$

Graph Convolutional Networks (GCNs)

Simplified Truncated Chebyshev polynomials based Graph Convolution:

$$\begin{aligned} h_\theta * X &\approx \theta_0 X + \theta_1 T_1(\tilde{X}) \\ &\approx \theta_0 X - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \\ &= \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) X \end{aligned}$$

$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

eigenvalues are in [0, 2]

Graph Convolutional Networks (GCNs)

Simplified Truncated Chebyshev polynomials based Graph Convolution:

$$
\begin{aligned}
h_\theta * X &\approx \theta_0 X + \theta_1 T_1(\tilde{X}) \\
&\approx \theta_0 X - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \\
&= \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) X
\end{aligned}
$$

$$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

eigenvalues are in [0, 2]

$\Rightarrow$

$$\tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}}$$

$$\tilde{D}_{ii} = \sum_j (A + I)_{ij}$$

eigenvalues are in [-1, 1]

Graph Convolutional Networks (GCNs)

Simplified Truncated Chebyshev polynomials based Graph Convolution:

$$\begin{aligned} h_\theta * X &\approx \theta_0 X + \theta_1 T_1(\tilde{X}) \\ &\approx \theta_0 X - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \\ &= \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) X \end{aligned}$$

$$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

eigenvalues are in [0, 2]

⇨

$$\tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}}$$

$$\tilde{D}_{ii} = \sum_j (A + I)_{ij}$$

eigenvalues are in [-1, 1]

Final Form of Graph Convolution: $h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}} X$

Graph Convolutional Networks (GCNs)

Graph convolution in GCNs for 1D graph signal:

$$h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}X$$

Graph Convolutional Networks (GCNs)

Graph convolution in GCNs for 1D graph signal:

$$h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}X$$

Generalize to multi-input and multi-output convolution:

$$\begin{aligned} h_W * X &\approx \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}XW \\ &= \tilde{L}XW \end{aligned}$$

Graph Convolutional Networks (GCNs)

Graph convolution in GCNs for 1D graph signal:

$$h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}X$$

Generalize to multi-input and multi-output convolution:

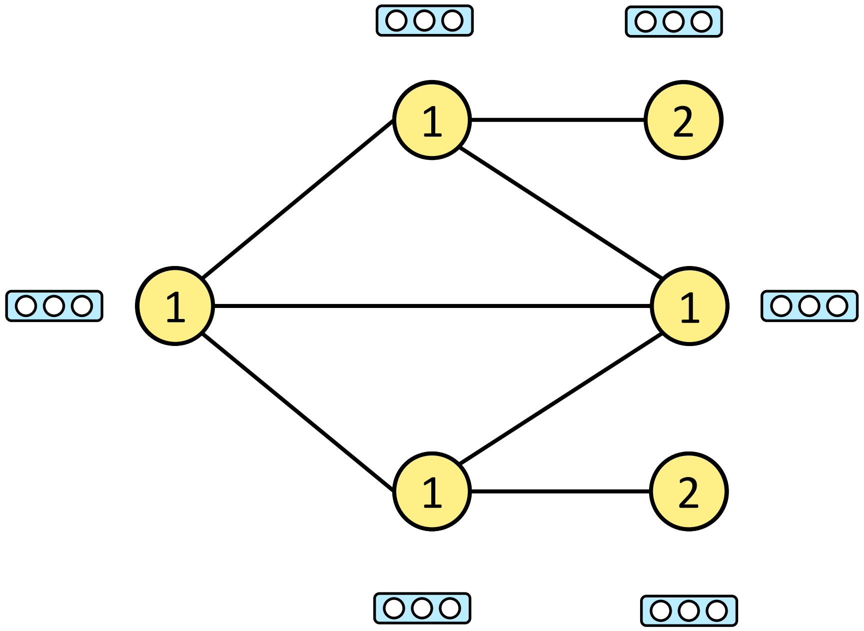
$$\begin{aligned} h_W * X &\approx \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}XW \\ &= \tilde{L}XW \end{aligned}$$

Add nonlinearity:

$$\sigma\left(h_W * X\right) \approx \sigma\left(\tilde{L}XW\right)$$

Graph Convolutional Networks (GCNs)

Our Spectral Filters are Localized:



$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}$$

1-step Graph Convolution: $h_W * X \approx \tilde{L}XW$

2-step Graph Convolution: $h_{W_2} * h_{W_1} * X \approx \tilde{L}^2 X W_1 W_2$

⋮

Exponent of matrix power indicates how far the propagation is!

Graph Convolutional Networks (GCNs)

- We start with Chebyshev Polynomials which can represent any spectral filters (eigenvalues in [-1, 1])

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

Graph Convolutional Networks (GCNs)

- We start with Chebyshev Polynomials which can represent any spectral filters (eigenvalues in [-1, 1])

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

- Truncate the expansion at 1^{st} order for efficiency

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

Graph Convolutional Networks (GCNs)

- We start with Chebyshev Polynomials which can represent any spectral filters (eigenvalues in [-1, 1])

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

- Truncate the expansion at 1st order for efficiency

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

- Further simplification/approximation

$$h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}X$$

$$h_W * X \approx \tilde{L}XW$$

Graph Convolutional Networks (GCNs)

- We start with Chebyshev Polynomials which can represent any spectral filters (eigenvalues in [-1, 1])

$$h(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

- Truncate the expansion at 1st order for efficiency

$$h_\theta * X \approx \theta_0 X + \theta_1 T_1(\tilde{X})$$

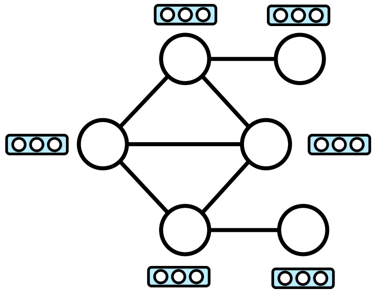
- Further simplification/approximation

$$h_\theta * X \approx \theta \tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}} X$$

$$h_W * X \approx \tilde{L} X W$$

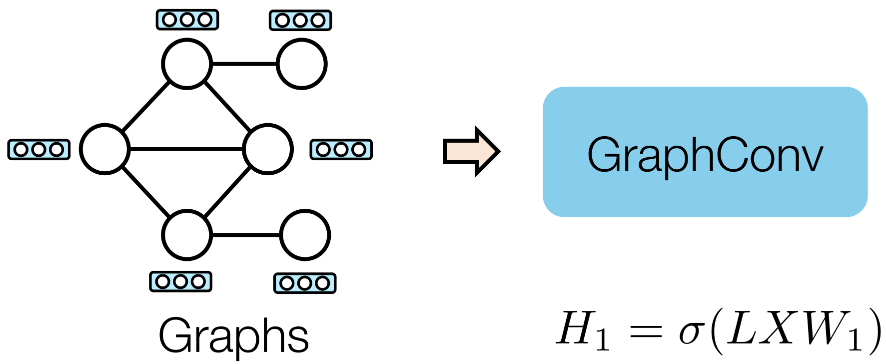
We can remedy the lost expressiveness by stacking multiple graph convolution layers!

Graph Convolutional Networks (GCNs)



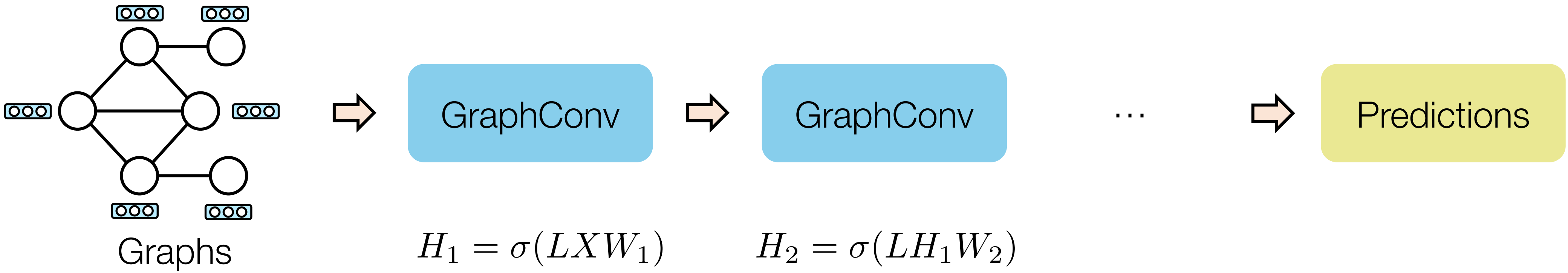
Graphs

Graph Convolutional Networks (GCNs)



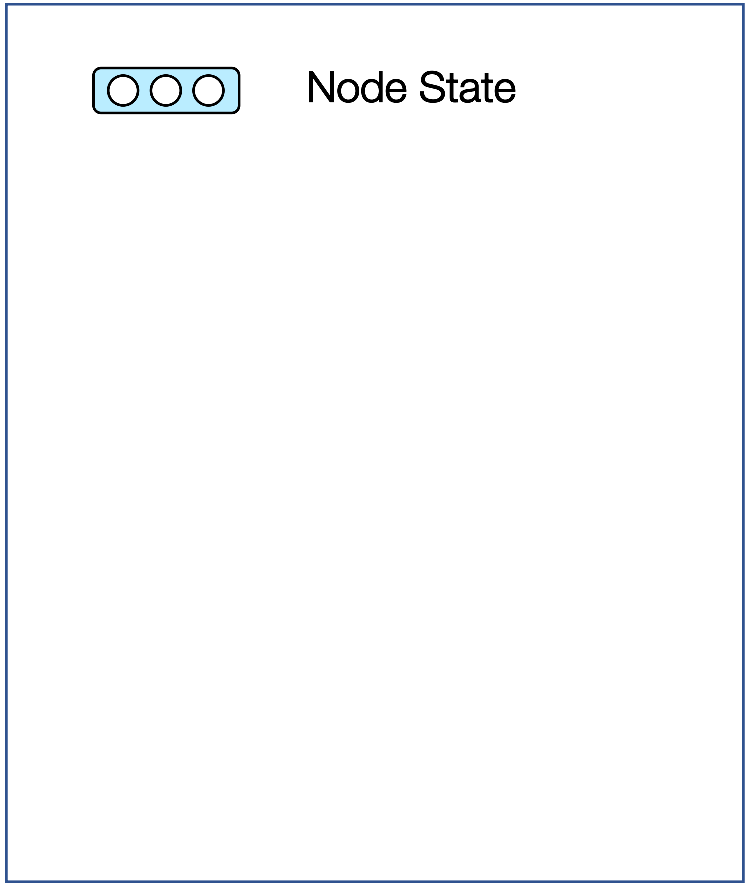
$$H_1 = \sigma(LXW_1)$$

Graph Convolutional Networks (GCNs)

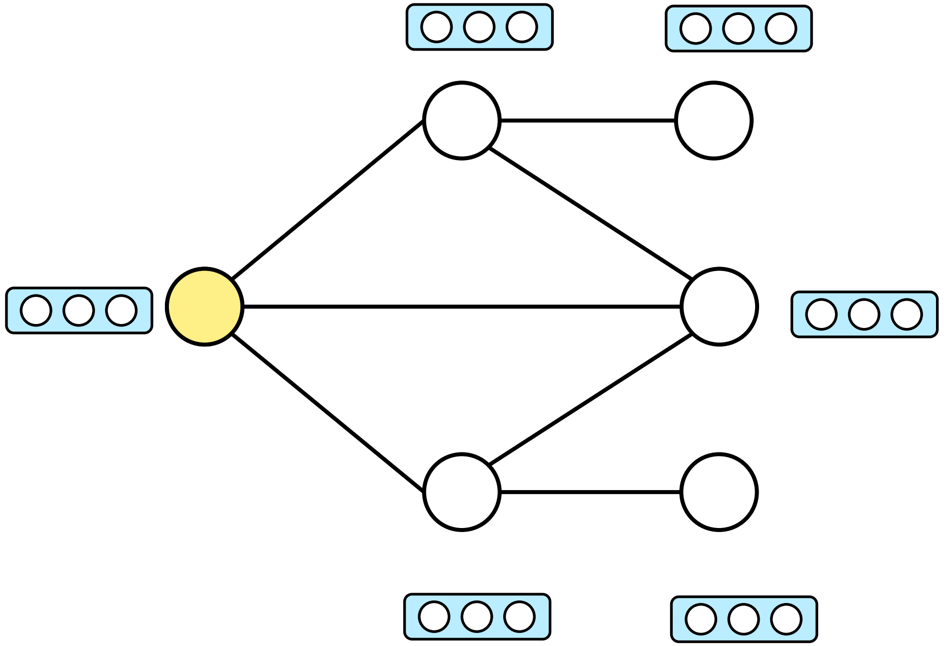


Message Passing GNNs

$\mathbf{h}_i^t$

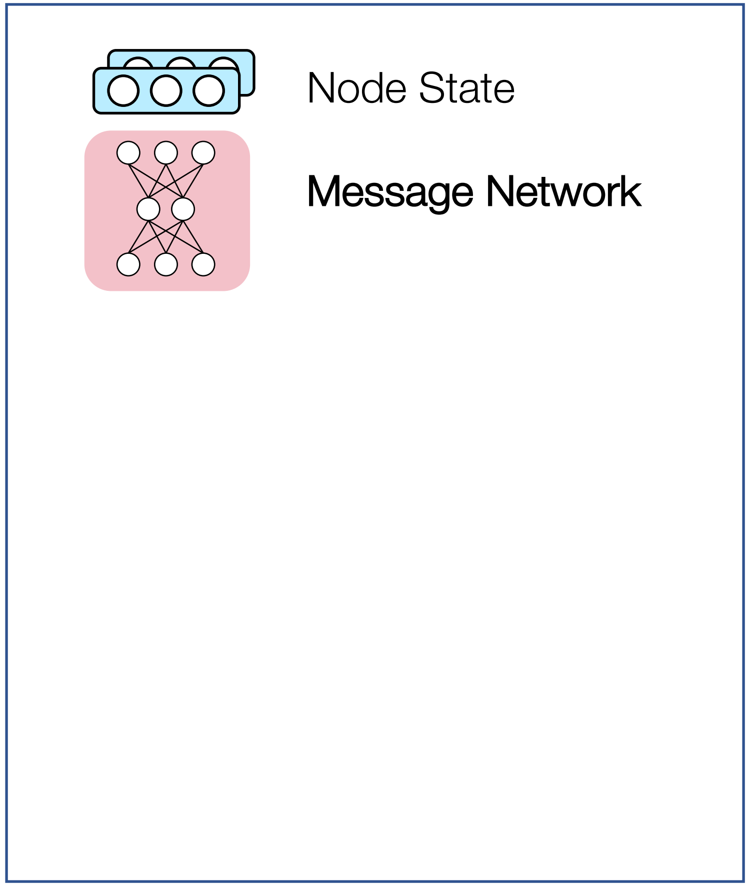


(t+1)-th message passing step/layer

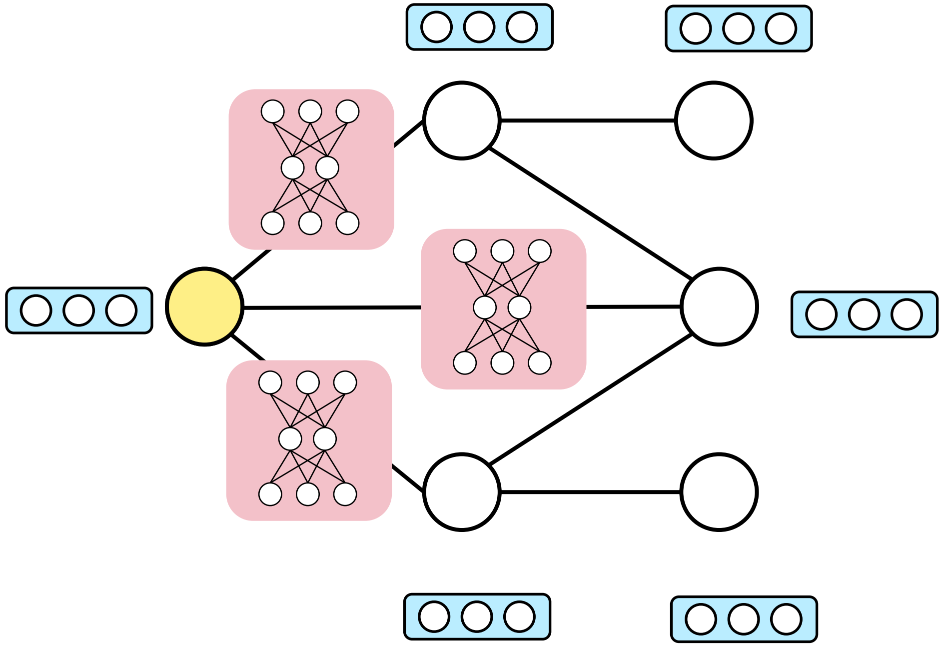


Message Passing GNNs

$\mathbf{h}_i^t \quad \mathbf{h}_j^t$



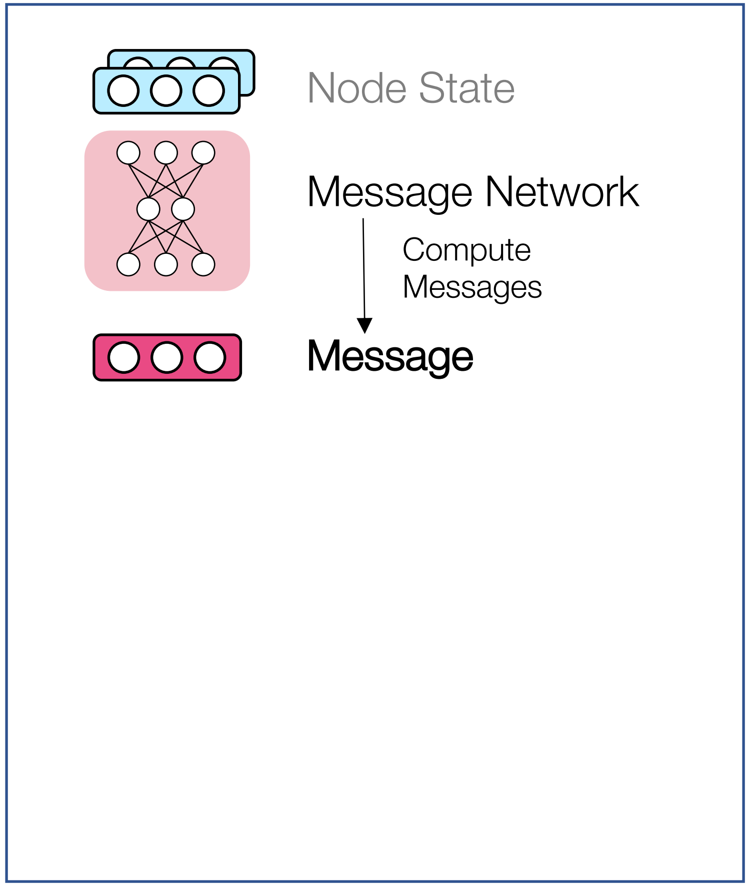
(t+1)-th message passing step/layer



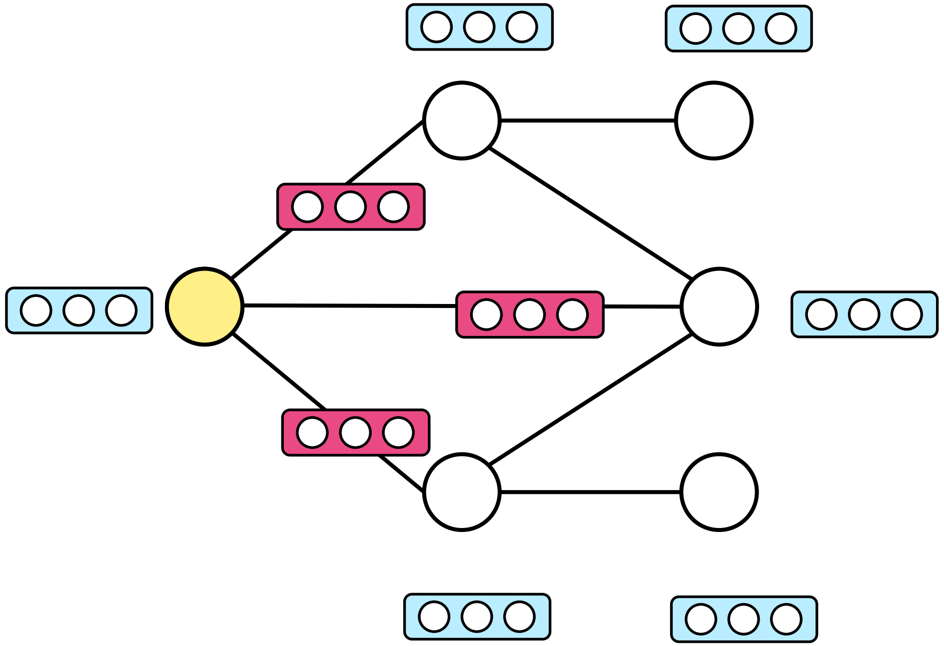
Message Passing GNNs

$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



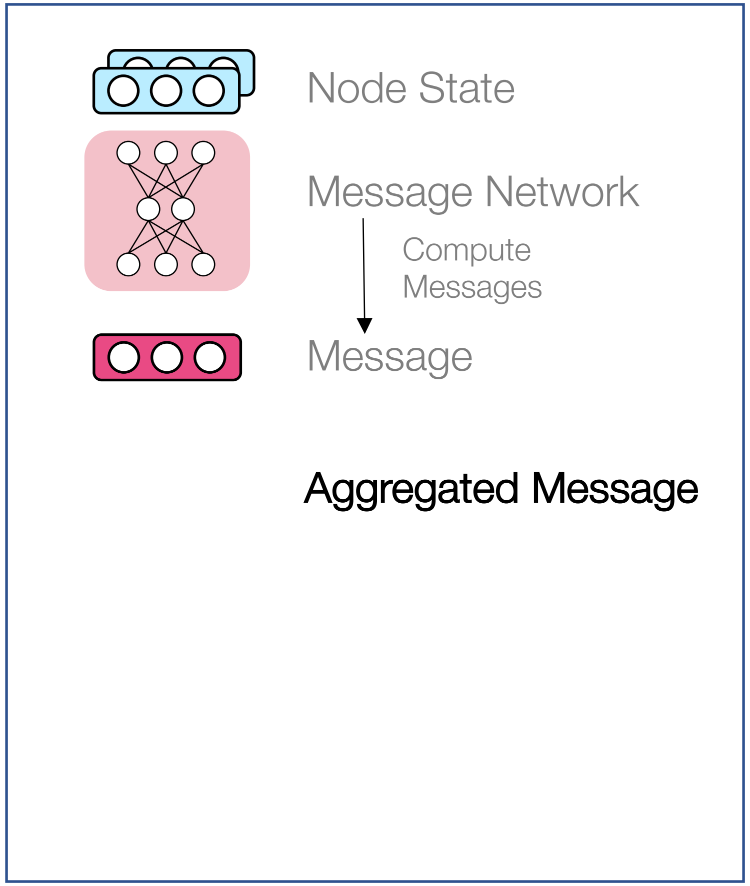
(t+1)-th message passing step/layer



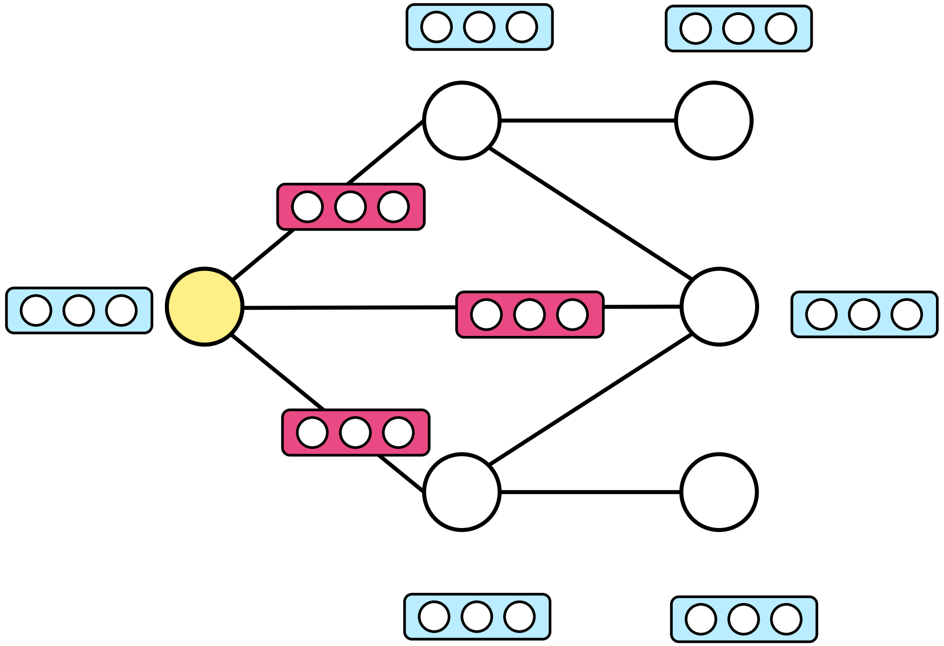
Message Passing GNNs

$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



(t+1)-th message passing step/layer

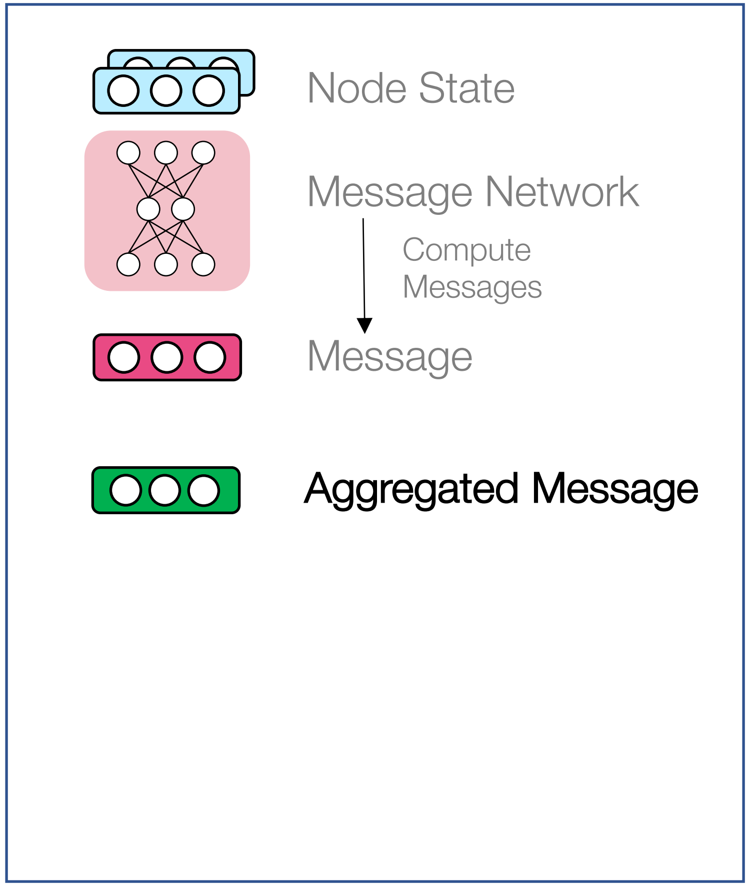


Message Passing GNNs

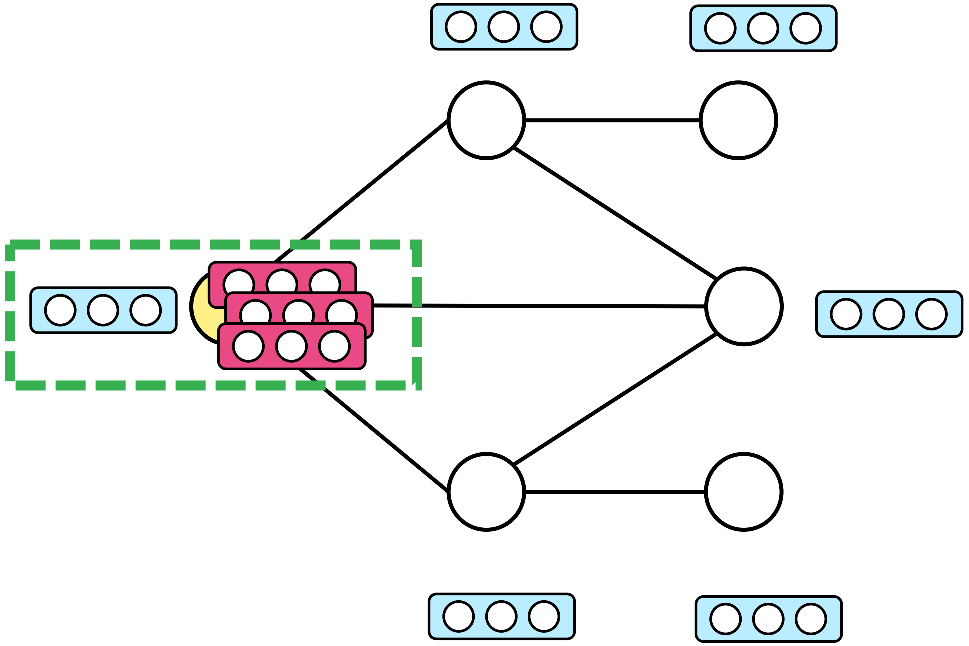
$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right)$$



(t+1)-th message passing step/layer

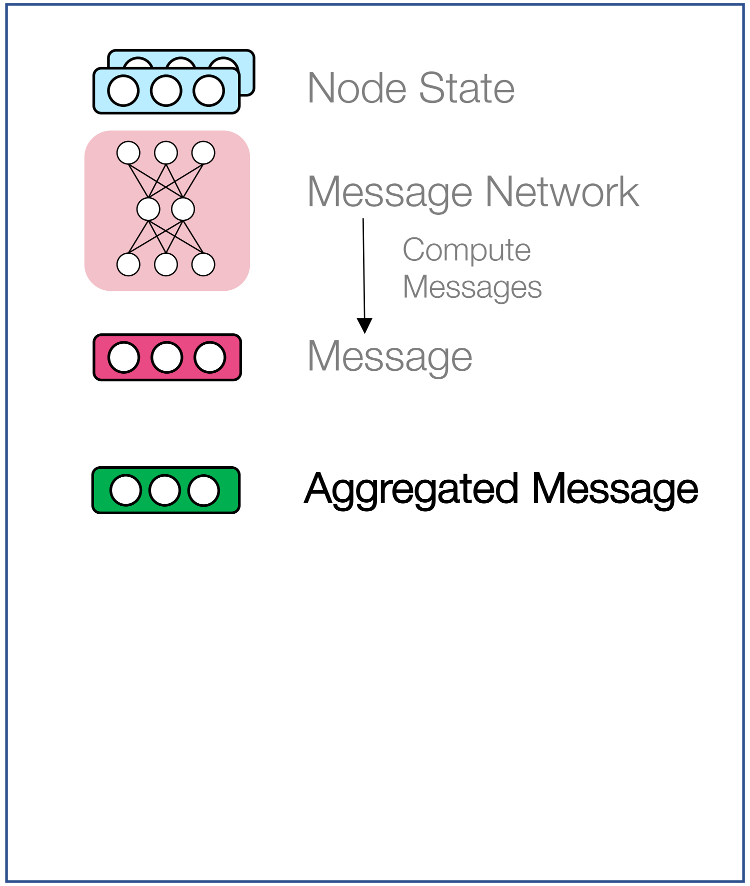


Message Passing GNNs

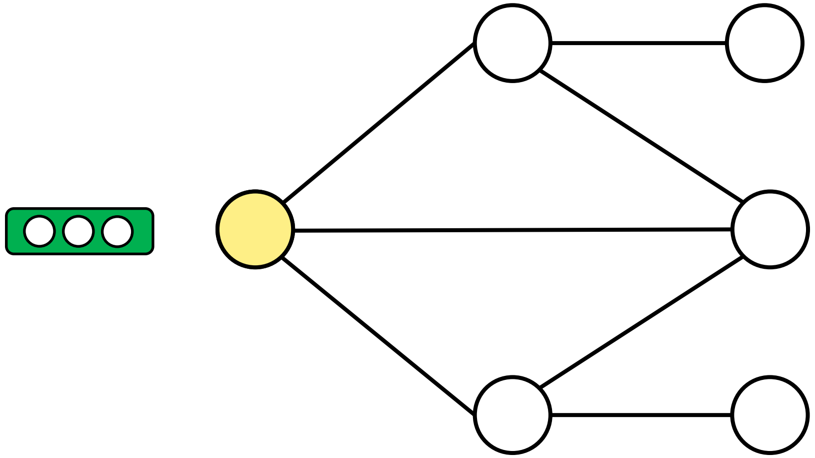
$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right)$$



(t+1)-th message passing step/layer

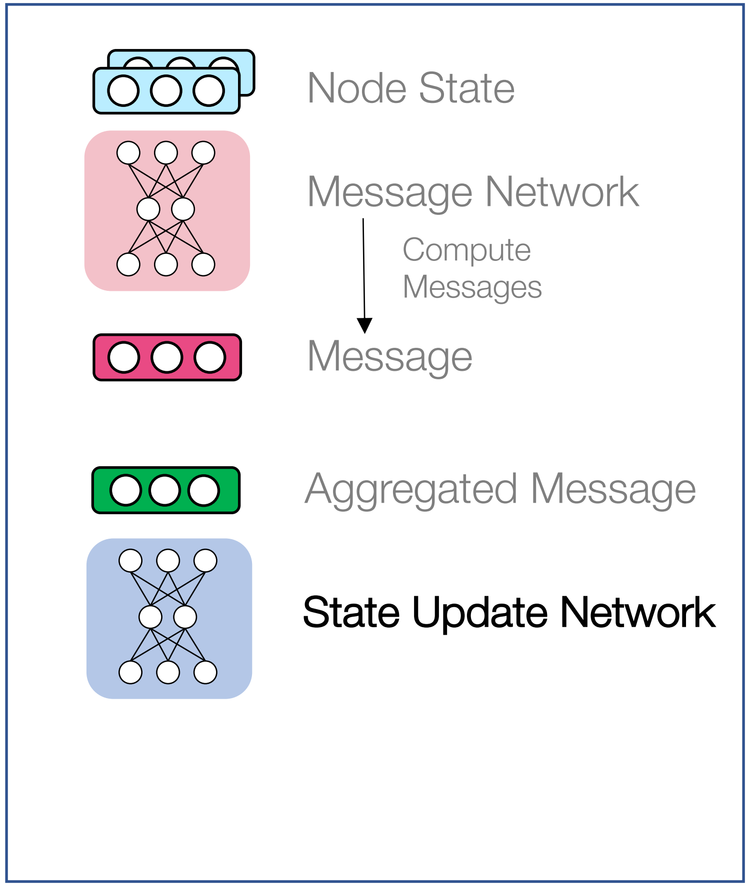


Message Passing GNNs

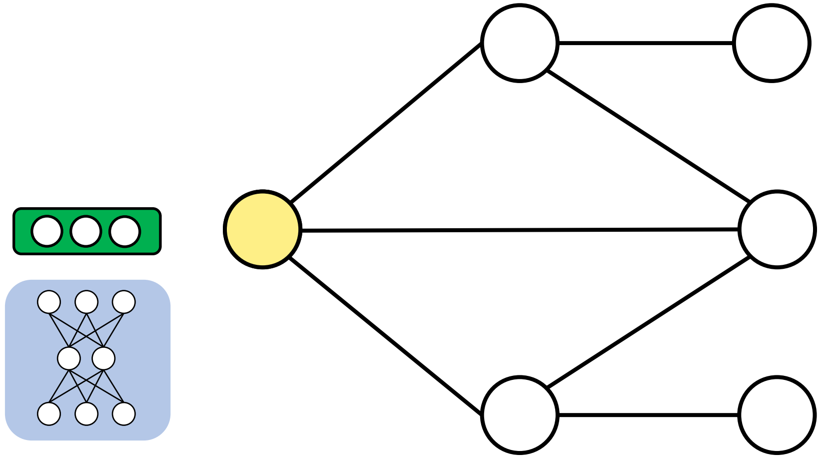
$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$

$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right)$



(t+1)-th message passing step/layer



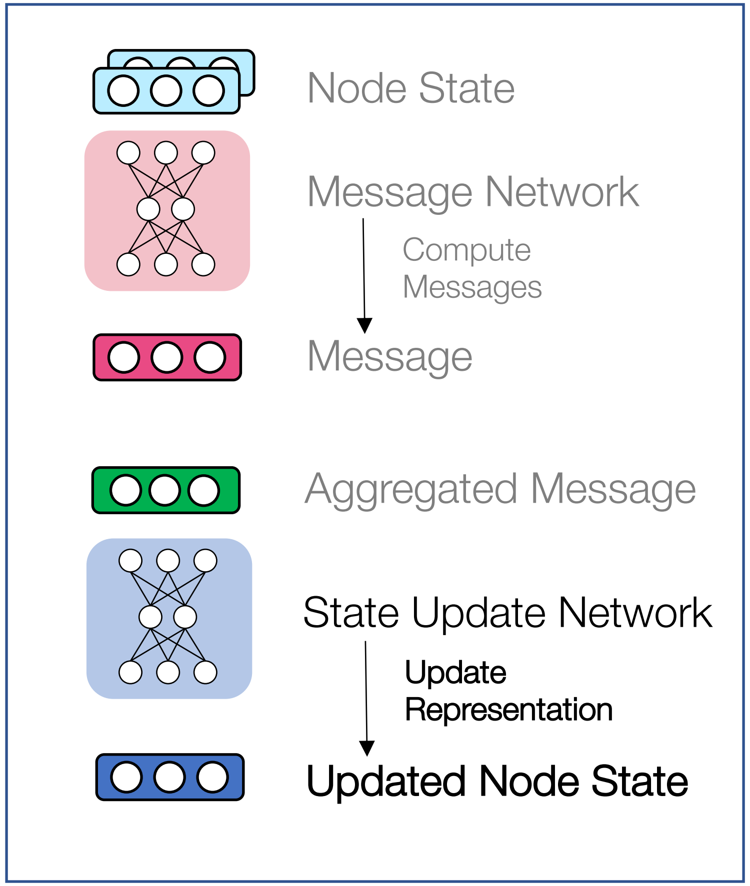
Message Passing GNNs

$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

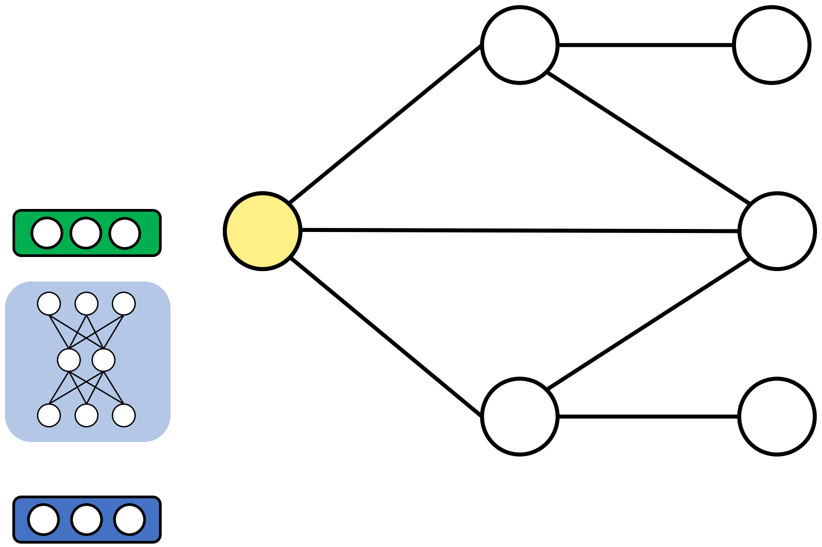
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right)$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



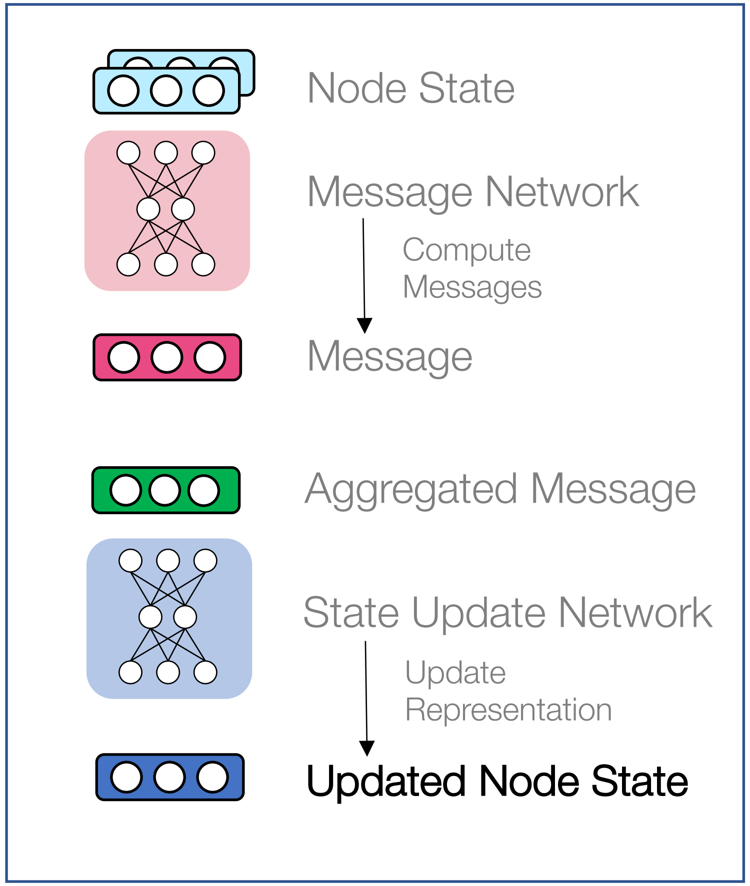
Message Passing GNNs

$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

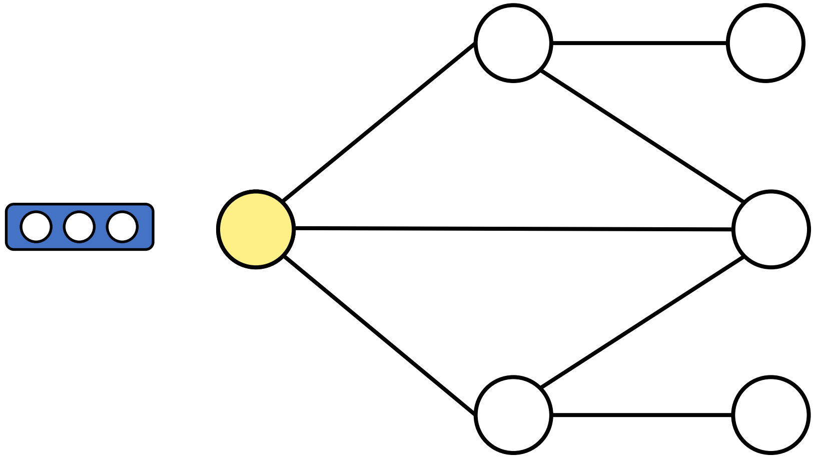
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right)$$

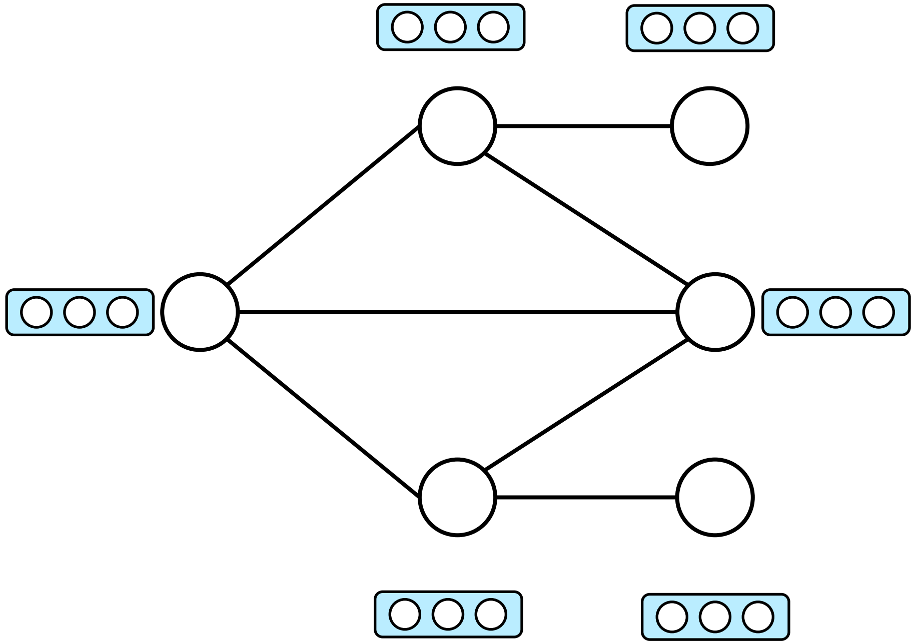
$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



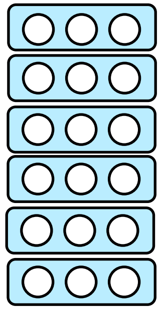
(t+1)-th message passing step/layer



GCNs are Message Passing Networks

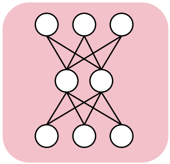


- Node State X

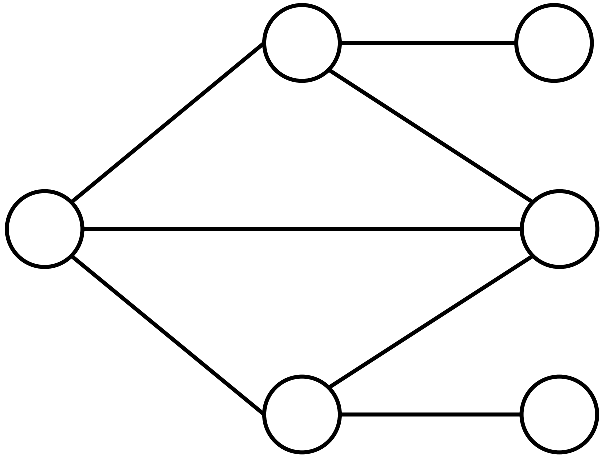


- Graph Laplacian

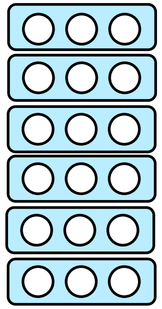
$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}$$



GCNs are Message Passing Networks

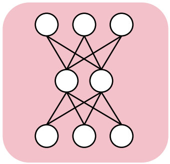


- Node State X



- Graph Laplacian

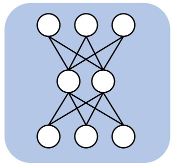
$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$$



- Aggregated Message

$$\tilde{L}X$$

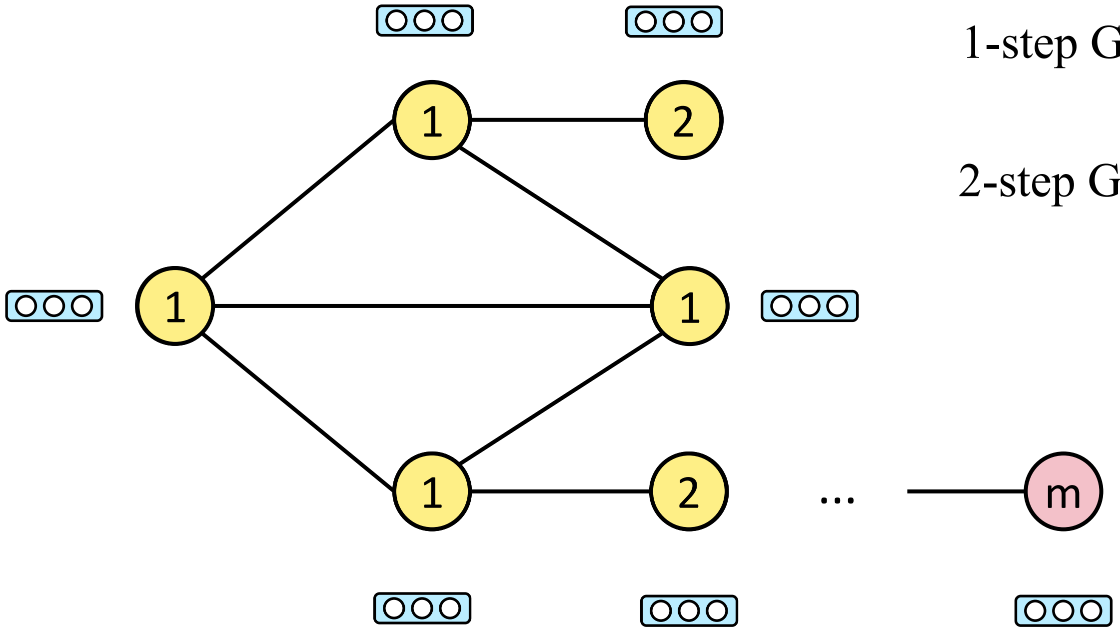
- State Update Network W



Revisit Spectral Filtering

Our Spectral Filters are Localized:

$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$$



1-step Graph Convolution: $h_W * X \approx \tilde{L}XW$

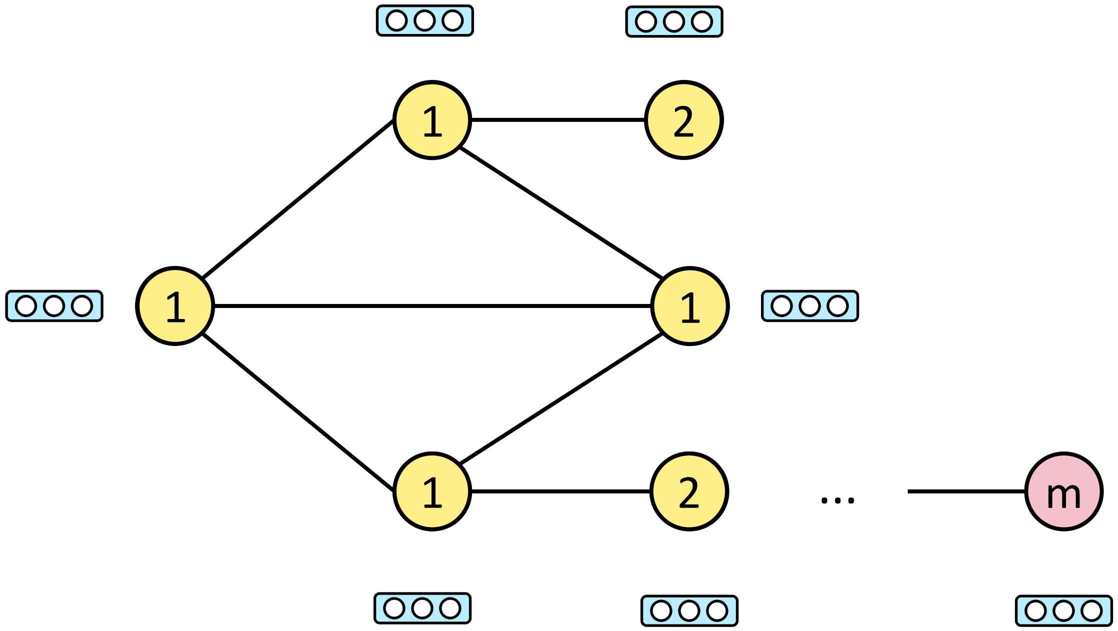
2-step Graph Convolution: $h_{W_2} * h_{W_1} * X \approx \tilde{L}^2 X W_1 W_2$

What if the graph diameter m is large?

Revisit Spectral Filtering

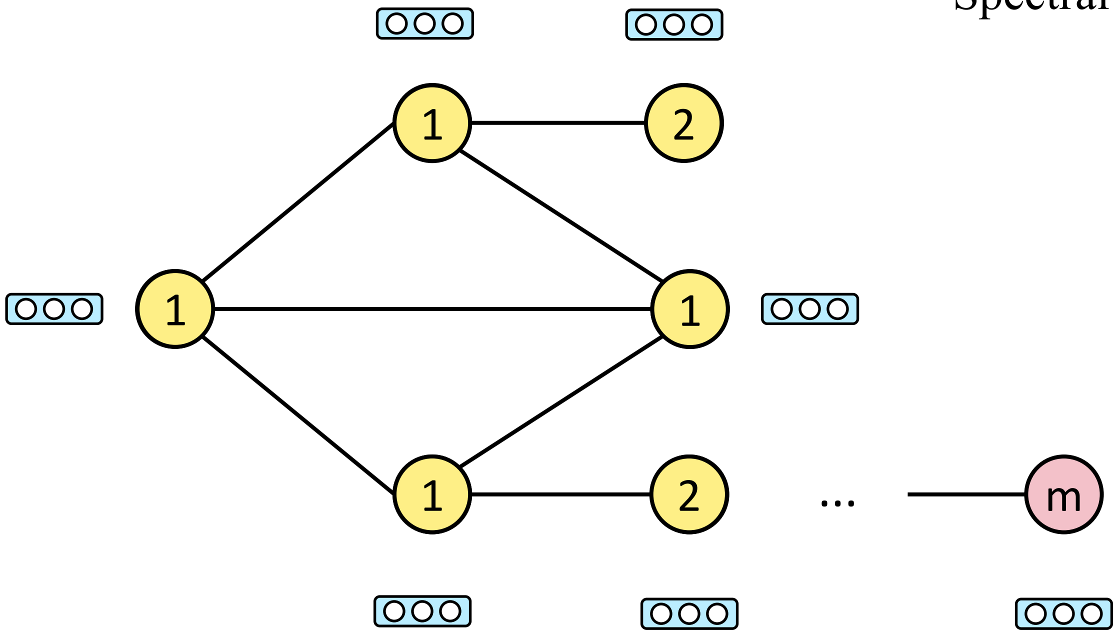
Our Spectral Filters are Localized:

m-step Graph Convolution: $h_W * X \approx \tilde{L}^m XW$



Revisit Spectral Filtering

Our Spectral Filters are Localized:



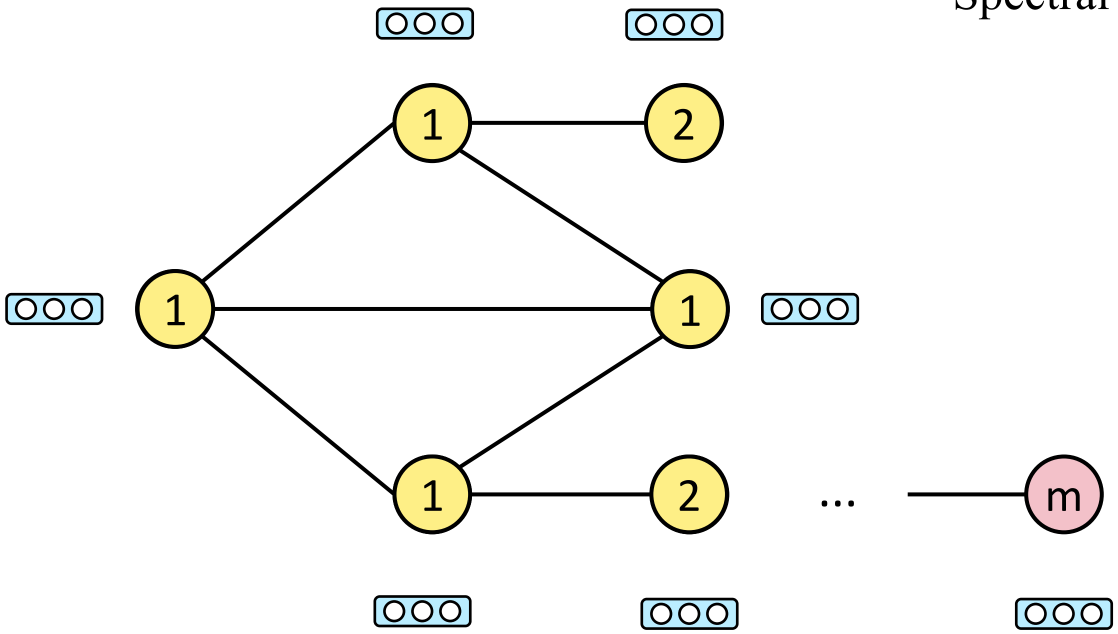
m-step Graph Convolution: $h_W * X \approx \tilde{L}^m X W$

Spectral Decomposition: $\tilde{L} = U \Lambda U^\top$

$$\tilde{L}^m = U \Lambda^m U^\top$$

Revisit Spectral Filtering

Our Spectral Filters are Localized:



m-step Graph Convolution: $h_W * X \approx \tilde{L}^m X W$

Spectral Decomposition: $\tilde{L} = U \Lambda U^\top$

$$\tilde{L}^m = U \Lambda^m U^\top$$

Cubic complexity O(N^3) !

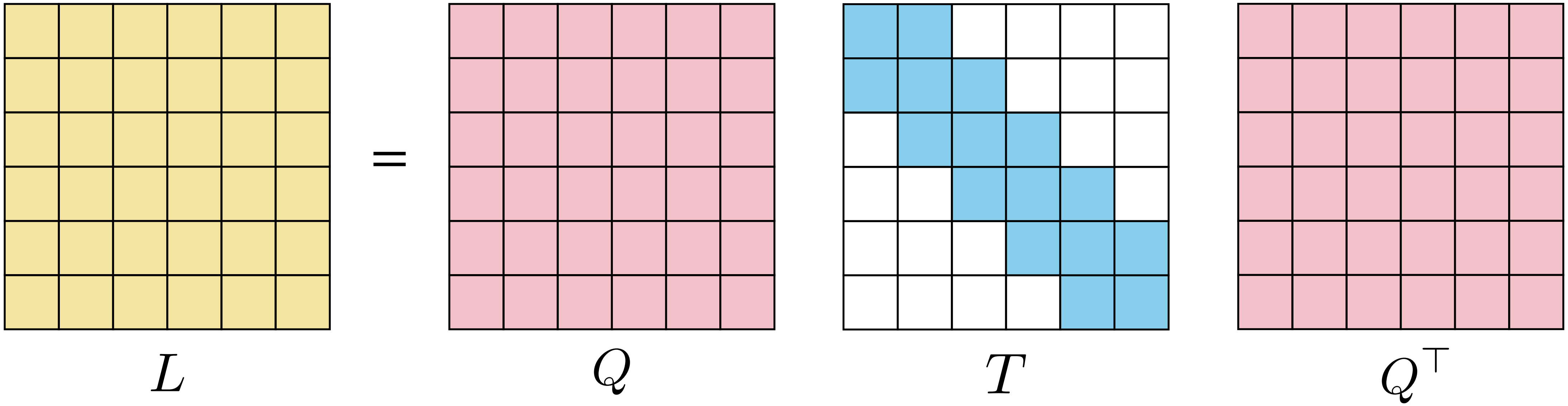
Lanczos Algorithm

Algorithm 1 : Lanczos Algorithm

Input: S, x, K, ϵ
Initialization: $\beta_0 = 0$, $q_0 = 0$, and $q_1 = x/\|x\|$
For $j = 1, 2, \ldots, K$:
    $z = Sq_j$
    $\gamma_j = q_j^\top z$
    $z = z - \gamma_j q_j - \beta_{j-1} q_{j-1}$
    $\beta_j = \|z\|_2$
    If $\beta_j < \epsilon$, quit
    $q_{j+1} = z/\beta_j$
$Q = [q_1, q_2, \cdots, q_K]$
Construct T following Eq. (2)
Eigen decomposition $T = BRB^\top$
Return $V = QB$ and R.

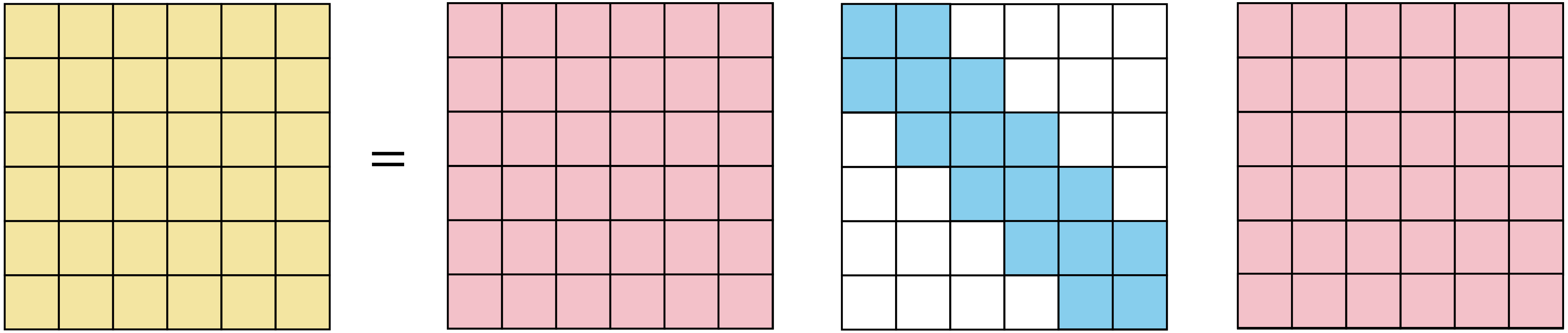
Lanczos Algorithm

Tridiagonal Decomposition $L = QTQ^\top$



Lanczos Algorithm

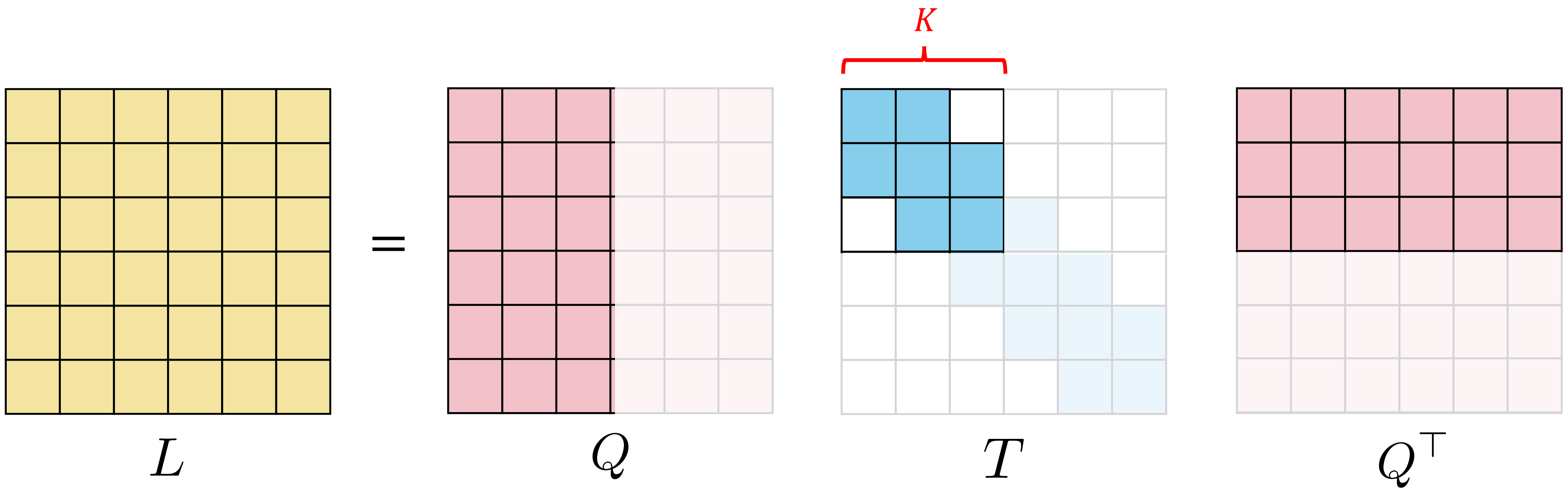
Tridiagonal Decomposition $L = QTQ^\top$



L Q T $Q^\top$

Lanczos Algorithm

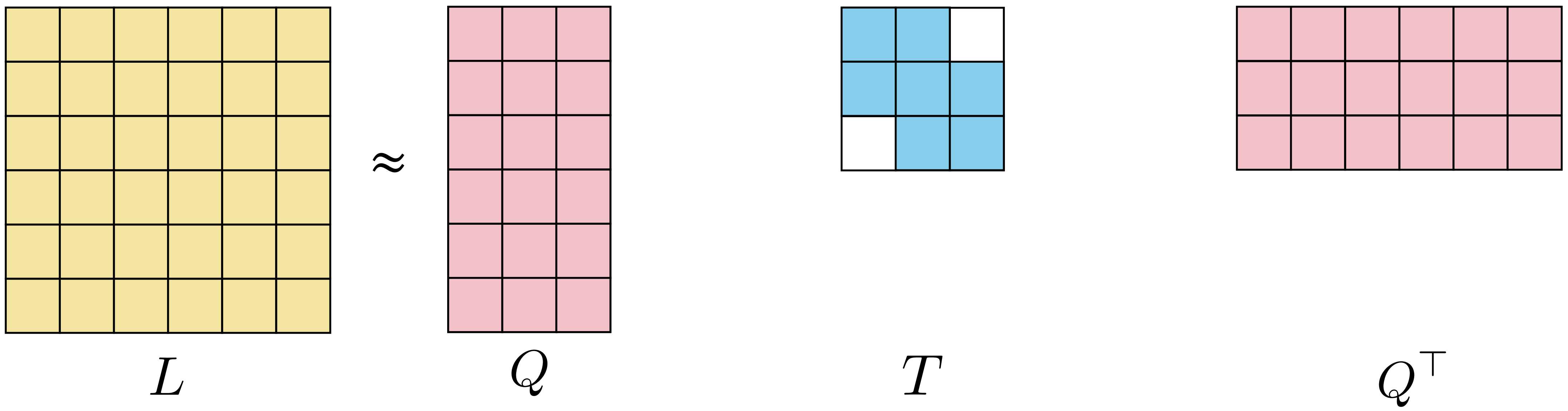
Tridiagonal Decomposition $L = QTQ^\top$



Lanczos Algorithm

Tridiagonal Decomposition $L = QTQ^\top$

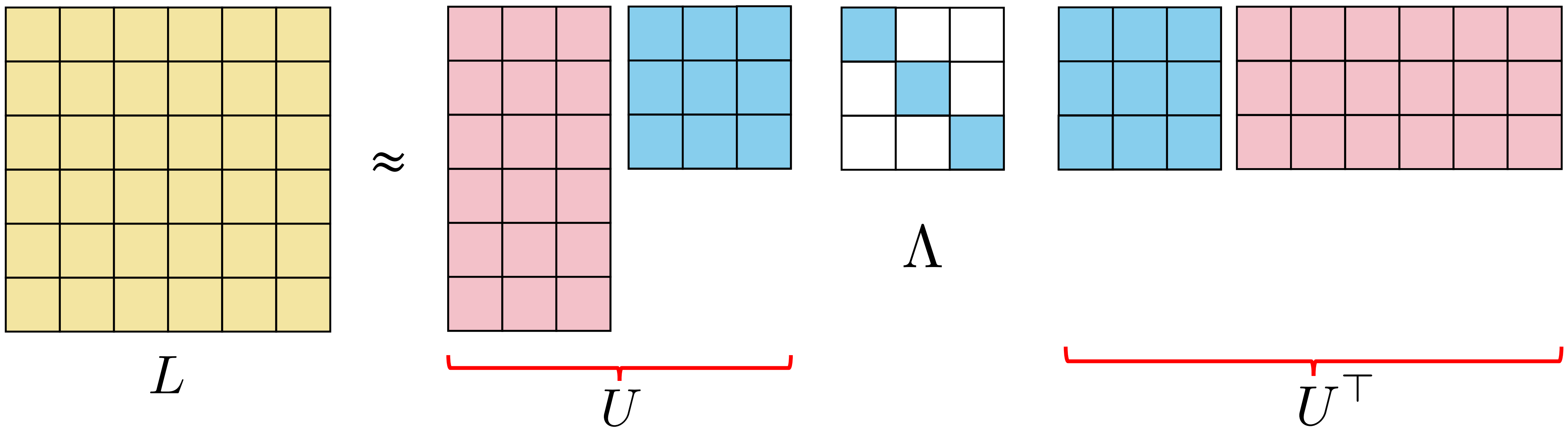
Low-rank approximation



Lanczos Algorithm

Tridiagonal Decomposition $L = QTQ^\top$

Low-rank approximation with **top K eigenpairs**



$O(N^3)$ → $O(KN^2)$

Multi-scale Graph Convolutional Networks

- m-step GraphConv (Prior Work) $H = L^m XW$

LanczosNet [9]:

- m-step GraphConv $H = U\Lambda^m U^\top XW$
- Learn Nonlinear Spectral Filter $H = U f_\theta(\Lambda^m) U^\top XW$
- Learning Graph Kernel / Metric $L_{ij} \propto \exp\left(-\|(X_i - X_j)M\|^2\right)$

References

[1] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G., 2008. The graph neural network model. IEEE transactions on neural networks, 20(1), pp.61-80.

[2] Goller, C. and Kuchler, A., 1996, June. Learning task-dependent distributed representations by backpropagation through structure. In Proceedings of International Conference on Neural Networks (ICNN'96) (Vol. 1, pp. 347-352). IEEE.

[3] Ackley, D.H., Hinton, G.E. and Sejnowski, T.J., 1985. A learning algorithm for Boltzmann machines. Cognitive science, 9(1), pp.147-169.

[4] Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A. and Vandergheynst, P., 2013. 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), pp.83-98.

[5] Ortega, A., Frossard, P., Kovačević, J., Moura, J.M. and Vandergheynst, P., 2018. Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5), pp.808-828.

[6] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A. and Vandergheynst, P., 2017. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4), pp.18-42.

[7] Hammond, D.K., Vandergheynst, P. and Gribonval, R., 2011. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2), pp.129-150.

[8] Kipf, T.N. and Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[9] Liao, R., Zhao, Z., Urtasun, R. and Zemel, R.S., 2019. Lanczosnet: Multi-scale deep graph convolutional networks. arXiv preprint arXiv:1901.01484.

Questions?