Spectral Basis of GNNs

Ng Yen Kaow

GNN history

1997	Sperduti and Starita Supervised neural networks for the classification of structures		
		LeNet-5	1998
2005	Gori *et al.* A new model for learning in graph domains		
2009	Scarselli *et al.* The graph neural network model		
	Hammond *et al.* Wavelets on graph via spectral graph theory		
	Micheli Neural networks for graph: A contextual constructive approach		
2010	Gallicchio and Micheli Graph echo state networks		
		AlexNet (U of T) wins ILSVRC	2012
2013	Shuman *et al.* The emerging field of signal processing on graphs		2013
	Bruna *et al.* Spectral networks and locally-connected networks on graphs		
		ZFNet (NYU) wins ILSVRC	
		GoogLeNet and VGGNet wins ILSVRC	2014
2015	Henaff *et al.* Deep convolutional networks on graph-structured data		2015
		ResNet wins ILSVRC	
2016	Defferrard *et al.* Convolutional neural networks on graphs with fast localized spectral filtering		
	Kipf and Welling Semi-supervised classification with graph convolutional networks		
	Atwood and Towsley Diffusion-convolutional neural networks		
	Niepert *et al.* Learning convolutional neural networks for graphs		
2017	Gilmer *et al.* Neural message passing for quantum chemistry		
2018	Battaglia *et al.* Relational inductive biases, deep learning, and graph networks		

RecGNN
Graph Fourier Transform
Spectral ConvGNN
Spatial ConvGNN

GNN history (significant eras)

1997 Sperduti and Starita Supervised neural networks for the classification of structures

LeN t-5 1998

2005 Gori *et al.* A new model for learning in graph domains

2009 Scarselli *et al.* The graph neural network model

Hammond *et al.* Wavelets on graph via spectral graph theory

Micheli Neural networks for graph: A contextual constructive approach

- Theory of spectral domain filters
- Idea of graph-based convolution

2010 Gallicchio and Micheli Graph echo state networks

AlexNet (U of T) wins ILS RC 2012

2013 Shuman *et al.* The emerging field of signal processing on graphs 2013

Bruna *et al.* Spectral networks and locally-connected networks on graphs

ZFNet (

GoogLeNet and V

- Spectral domain filters as NNs and their approximation techniques

2015 Henaff *et al.* Deep convolutional networks on graph-structured data

R

2016 Defferrard *et al.* Convolutional neural networks on graphs with fast localized spectral ering

Kipf and Welling Semi-supervised classification with graph convolutional networks

Atwood and Towsley Diffusion-convolutional neural networks

Niepert *et al.* Learning convolutional neural networks for graphs

RecGNN

- Adding up neighbors is all you need

2017 Gilmer *et al.* Neural message passing for quantum chemistry

2018 Battaglia *et al.* Relational inductive biases, deep learning, and graph networks

GNN history (the people behind)

1997 Sperduti and Starita Supervised neural networks for the classification of structures

LeCun LeNet-5 1998

2005 Gori *et al.* A new model for learning in graph domains *(first use of the term GNN)*

2009 Scarselli *et al.* The graph neural network model

Hammond *et al.* Wavelets on graph via spectral graph theory

Micheli Neural networks for graph: A contextual constructive approach

2010 Gallicchio and Micheli Graph echo state networks

Sutskever+Hinton AlexNet (U of T) wins ILSVRC 2012

2013 Shuman *et al.* The emerging field of signal processing on graphs 2013

LeCun Bruna *et al.* Spectral networks and locally-connected networks on graphs

LeCun, sort of ZFNet (NYU) wins ILSVRC

LeCun

Google GoogLeNet and VGGNet wins ILSVRC 2014

2015 Henaff *et al.* Deep convolutional networks on graph-structured data 2015

ResNet wins ILSVRC Microsoft

2016 Defferrard *et al.* Convolutional neural networks on graphs with fast localized spectral filtering

Google Kipf and Welling Semi-supervised classification with graph convolutional networks *(GCN)*

Atwood and Towsley Diffusion-convolutional neural networks

Niepert *et al.* Learning convolutional neural networks for graphs

2017 Gilmer *et al.* Neural message passing for quantum chemistry Google

2018 Battaglia *et al.* Relational inductive biases, deep learning, and graph networks Google

RecGNN
Graph Fourier Transform
Spectral ConvGNN
Spatial ConvGNN

Graph Fourier transform

- Let U be a eigenbasis of some Laplacian L
- Then $U^\top x$ is a projection of distribution x on eigenbasis U
  - $$U^\top x = \begin{bmatrix} \leftarrow & \mu_1 & \rightarrow \\ \leftarrow & \mu_2 & \rightarrow \\ & \vdots & \end{bmatrix} x = \begin{bmatrix} \mu_1^\top x \\ \mu_2^\top x \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix}}_{\text{dimension of Fourier space (= \#eigenvectors)}} = \dot{x}$$

    where $a_i = \mu_i x$ is the projection onto μ_i
  - The projected space is $\sum_i a_i \mu_i$

x and H will be used interchangeably here

Hammond *et al.* Wavelets on graphs via spectral graph theory, 2009

Graph Fourier transform

- Let U be a eigenbasis of some Laplacian L
- Then $U^\top x$ is a projection of distribution x on eigenbasis U
- An application of U would transform $\dot{x}$ back into x

$$U\dot{x} = \begin{bmatrix} \uparrow & \uparrow & \\ \mu_1 & \mu_2 & \dots \\ \downarrow & \downarrow & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \mu_{11}a_1 + \mu_{21}a_2 + \cdots \\ \mu_{12}a_1 + \mu_{22}a_2 + \cdots \\ \vdots \end{bmatrix}$$

$$= \mu_1 a_1 + \mu_2 a_2 + \cdots = \mu_1 \mu_1^\top x + \mu_2 \mu_2^\top x + \cdots$$

$$= \left(\sum_i \mu_i \mu_i^\top\right) x = Ix = x$$

Homework: prove $\sum_i \mu_i \mu_i^\top = I$

Graph Fourier transform

- Let U be a eigenbasis of some Laplacian L
- Then $U^\top x$ is a projection of distribution x on eigenbasis U
- An application of U would transform $\dot{x}$ back into x, $U(\dot{x}) = U(U^\top x) = x$ (obvious since $UU^\top = I$)
- Denote $U^\top x$ as $F(x)$ and $U\dot{x}$ as $F^{-1}(\dot{x})$

Graph Fourier transform

- A convolution of x in the Fourier domain of a graph G is $x * g = F^{-1}\big(F(x) \odot F(g)\big) = U(U^\top x \odot U^\top g)$ where U is the eigenbasis of some Laplacian of G, g is some filter that works on the eigenbasis U, and $\odot$ is the element-wise (Hadamard) product
- Suppose $U^\top g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \end{bmatrix}$. Let $g_\theta = \mathrm{diag}(U^\top g) = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix}$

  Then we can write $x * g = U g_\theta U^\top x$ (shown below)
  - Each g_i weights the significance of the eigenvector μ_i
  - g_θ is to be inferred
  - This inference task results in the spectral GNNs

$$U^\top x \odot U^\top g = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix} \odot \begin{bmatrix} g_1 \\ g_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 g_1 \\ a_2 g_2 \\ \vdots \end{bmatrix}$$

$$g_\theta U^\top x = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 g_1 \\ a_2 g_2 \\ \vdots \end{bmatrix}$$

Henaff *et al.* Deep Convolutional Networks on Graph-Structured Data, 2013

Spectral GNN

- **The spectral GNN task of learning a function f and filter g for graph G, is to infer f and the coefficients g_1, g_2, …, such that for each x, $f(U g_\theta U^\top x)$ matches the desired output**
  - These GNNs work in the spectral domain as opposed to the spatial domain of the graph
  - g_θ is to be independent of the eigenvectors U
    - That is, $g_\theta(L) = g_\theta(U\Lambda U^\top) = U g_\theta(\Lambda) U^\top x$ where L is some Laplacian for G
    - Of course, g_θ may turn out to be independent of Λ
      - In which case, g_θ is inferred solely from the examples
- In spectral GNNs we learn which eigenvectors to use from examples in a supervised learning
  - In spectral clustering we take the eigenvectors of the slowest growth (hence more "global") and perform unsupervised learning with those vectors

Chebyshev approximation for U

- However, computing U is $O(N^3)$ and computing $U^\top x$ is $O(N^2)$ ⇒ expensive
- Approximate g_θ with Chebyshev polynomials

$$g_\theta(\Lambda) \approx g_{\theta'}(\Lambda) = \sum_{i=0}^{K} \theta_i' \, T_i(\tilde{\Lambda})$$

where

- $\tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - I$ ($\lambda_{\max}$ is the largest eigenvalue)
- $\theta' \in \mathbb{R}^K$ are Chebyshev coefficients, and
- The polynomials $T_i(x)$ are computed with a recurrence relation
  - $T_0(x) = 1, T_1(x) = x$ (base case)
  - $T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x)$
- K is the number of expansion terms

Defferrard *et al.* Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, 2016

Chebyshev approximation for U

- However, computing U is $O(N^3)$ and computing $U^\top x$ is $O(N^2) \Rightarrow$ expensive
- Approximate g_θ with Chebyshev polynomials

$$g_\theta(\Lambda) \approx g_{\theta'}(\Lambda) = \sum_{i=0}^{K} \theta_i' \, T_i(\tilde{\Lambda})$$

- Then

$$x * g_\theta = U g_\theta U^\top x \approx U \left(\sum_{i=0}^{K} \theta_i' \, T_i(\tilde{\Lambda}) \right) U^\top x$$

- Since $UT(\tilde{\Lambda})U^\top = \frac{2}{\lambda_{\max}} U \Lambda U^\top - IUU^\top = \frac{2}{\lambda_{\max}} L - I$

Write $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$ and $\boxed{x * g_\theta \approx \sum_{i=0}^{K} \theta_i' \, T_i(\tilde{L}) \, x}$

Chebyshev approximation

Chebyshev approximation for U

- T_n, the nth order coefficient of the Chebyshev polynomials **of the first kind**, is

$$T_n(\cos\theta) = \cos n\theta$$

  - The coefficients can be obtained using the recurrence relation

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos\theta\cos n\theta$$
$$\Rightarrow T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x)$$

0th order
  - $\cos 0\theta = 1 \quad \Rightarrow T_0(x) = 1$

1st order
  - $\cos 1\theta = \cos\theta \quad \Rightarrow T_1(x) = x$

2nd order
  - $\cos 2\theta = 2\cos^2\theta - 1 \quad \Rightarrow T_2(x) = 2x^2 - 1$

3rd order
  - $\cos 3\theta = 4\cos^3\theta - 3\cos\theta \quad \Rightarrow T_3(x) = 4x^3 - 3x$

Chebyshev approximation for U

- Chebyshev approximation has $x * g_\theta \approx \sum_{i=0}^{K} \theta_i' \, T_i(\tilde{L}) \, x$
- To compute $T_i(\tilde{L})$, use the Chebyshev recurrence

$$T_0(\tilde{L}) = 1,\ T_1(\tilde{L}) = \tilde{L},\ T_{n+1}(\tilde{L}) = 2\,\tilde{L}\,T_n(\tilde{L}) - T_{n-1}(\tilde{L})$$

- Denote $\bar{x}_k = T_k(\tilde{L})\,x$, this becomes

$$\bar{x}_{n+1} = 2\,\tilde{L}\,\bar{x}_n - \bar{x}_{n-1}$$ (or $\bar{x}_n = 2\,\tilde{L}\,\bar{x}_{n-1} - \bar{x}_{n-2}$)

- Then, $x * g_\theta \approx \sum_{i=0}^{K} \theta_i' \; T_i(\tilde{L})\,x = [\theta_0' \quad \ldots \quad \theta_K'] \begin{bmatrix} \bar{x}_0 \\ \vdots \\ \bar{x}_K \end{bmatrix}$
  - Can be computed in $O(K|E|)$ time from $\tilde{L}$
- Precompute the K vectors $\bar{x}_0, \ldots, \bar{x}_K$, with the recurrence relation, and learn the scalars $\theta_0', \ldots, \theta_K'$

$K = 1$ (GCN) approximations

- Chebyshev approximation has $x * g_\theta \approx \sum_{i=0}^{K} \theta_i' T_i(\tilde{L})\, x$
- GCN takes $K = 1$ to obtain
$$x * g_{\theta'} \approx \theta_0' x + \theta_1' \tilde{L} x = \theta_0' x + \theta_1' \left(\frac{2}{\lambda_{\max}} L - I_N \right) x$$
- Since $\lambda_{\max} = 2$ we get $x * g_{\theta'} \approx \theta_0' x + \theta_1' (L - I_N) x$
  - θ_0' and θ_1' are parameters to be learned
- On the unweighted normalized Laplacian $L = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2} A D^{-1/2}$, this becomes
$$x * g_{\theta'} = \theta_0' x - \theta_1' D^{-1/2} A D^{-1/2} x$$
- Further constraint the number of parameters by letting $\theta_0' = -\theta_1' = \theta$
$$\boxed{x * g_{\theta'} = \theta\left(I + D^{-1/2} A D^{-1/2}\right) x}$$

GCN 1st order approximation

$K = 1$ (GCN) approximations

- However, since $L = I - D^{-1/2}AD^{-1/2}$

  $\Rightarrow x * g_{\theta'} = \theta\left(I + D^{-1/2}AD^{-1/2}\right)x = \theta(2I - L)x$
- Then, multiple applications of $\theta(2I - L)$ would result in

$$\theta^k(2I - L)^k x = \theta^k U(2 - \Lambda)^k U^\top x$$

where Λ/U are the eigenvalues/eigenvectors for L

(GCN places non-linear functions between layers which we ignore in this derivation)

  - Since L has eigenvalues in $[0, \lambda_{\max}]$ (where $\lambda_{\max} \leq 2$ is the largest eigenvalue of L)

    $\Rightarrow (2 - \Lambda)^k$ has range of $[(2 - \lambda_{\max})^k, 2^k]$

    $\Rightarrow$ Exponentially large spectral coefficients at higher k
- Solution: Let $\hat{A} = A + I$ (augmentation) and normalize $\hat{A}$ (renormalization)

Augmented adjacency matrix

That is, $\boxed{x * g_{\theta'} = \theta\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}x}$ where $\hat{D}_{ii} = \sum_i \hat{A}_{ij}$

How legit are GCN approximations

- Consider the two approximations of $x * g_\theta$ in GCN

  1. $S_{\text{1-order}} = \theta\left(I + D^{-1/2}AD^{-1/2}\right)$, or
  2. $\hat{S}_{\text{adj}} = \theta\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}$ $(\hat{A} = A + I)$

  where θ is a scalar to be learned

- Evaluate how well they approximate $x * g_\theta$ in the case that $g_\theta = \text{diag}(\Lambda)$, that is,

$$x * g_\theta = (Ug_\theta U^\top)x = (U\Lambda U^\top)x = Lx$$

- First, letting $\theta_0' = -\theta_1'$ (case of $S_{\text{1-order}}$) or $\theta_0' = \theta_1'$ would result in $x * g_{\theta'}$ having the same eigenvectors as L, that is,
  - $\theta_0' = -\theta_1' \Rightarrow x * g_{\theta'} = \theta(2I - L)x$
    $\Rightarrow$ same eigenvectors but eigenvalues become $2 - \lambda$
  - $\theta_0' = \theta_1' \Rightarrow x * g_{\theta'} = \theta Lx$
    $\Rightarrow$ same eigenvalues/ eigenvectors

How legit are GCN approximations

- Use the Karate club graph for L
- Comparison of eigenvectors/ eigenvalues

Filter	Eigenvalues	Eigenvector (corr. to smallest eigenvalue in L)
L	1.71, 1.61, 1.58, 1.57, ..., .39, .29, .13, 0	-.32, -.24, -.25, -.2, -.14, -.16, -.16, -.16, -.18, -.11, ..., -.14, -.14, -.11, -.16, -.14, -.16, -.16, -.2, -.28, -.33
$S_{\text{1-order}}$	2., 1.87, 1.71, 1.61, 1.39, ..., .5, .43, .42, .39, .29	.32, .24, .25, .2, .14, .16, .16, .16, .18, .11, ..., .14, .14, .11, .16, .14, .16, .16, .2, .28, .33
$\hat{S}_{\text{adj}}$	1., .9, .77, .7, .55, ..., -.21, -.22, -.27, -.31, -.42	.3, .23, .24, .19, .15, .16, .16, .16, .18, .13, ..., .15, .15, .13, .16, .15, .16, .16, .19, .26, .31

  - L and $S_{\text{1-order}}$ share the same eigenvectors
  - Eigenvectors of $\hat{S}_{\text{adj}}$ closely resembles those of L and $S_{\text{1-order}}$
- Evaluate MSE($S_{\text{1-order}}x$, Lx) and MSE($\hat{S}_{\text{adj}}x$, Lx) on randomly generated x
  - MSE($S_{\text{1-order}}x$, Lx) $= 0.159$ (obtained at $\theta \sim 0.1$)
  - MSE($\hat{S}_{\text{adj}}x$, Lx) $= 0.166$ (obtained at $\theta \sim 0.07$)
  - MSE(random vector, Lx) $= 0.413$
  - Better than random but lackluster performance due to differences in eigenvalues which were not remedied downstream

Matrices introduced so far

	Name	Eigenvalues range
A	Adjacency matrix	[-max(A), max(A)] (also see Bhunia *et al*. 2019)
$D - A$	Laplacian	[0, 2 max(A)]
$I - D^{-1/2}AD^{-1/2}$ (or $D^{-1/2}(D-A)D^{-1/2}$)	Normalized Laplacian	[0, 2]
$D^{-1/2}AD^{-1/2}$	Normalized adjacency matrix (Ng, Jordan, Weiss. 2001)	[-1, 1]
$I - D^{-1}A$	Random Walk Laplacian	(non-symmetric)
$I + D^{-1/2}AD^{-1/2}$	1st order approximation GCN (Kipf and Welling. 2016)	[0, 2]
$\hat{A} = I + A$	Augmented adjacency matrix	[-max$(\hat{A})$, max$(\hat{A})$]
$\hat{D} - \hat{A} = (D + I) - (A + I) = D - A$	(Augmented) Laplacian	[0, 2 max(A)]
$I - \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$	Normalized augmented Laplacian	[0, 2]
$\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$	Normalized augmented adjacency matrix GCN (Kipf and Welling. 2016)	[-1, 1]

Matrices introduced so far

	Name	Eigenvalues range
A	Adjacency matrix	$[-\max(A), \max(A)]$ (also see Bhunia *et al.* 2019)
$D - A$	Laplacian	$[0, 2\max(A)]$
$I - D^{-1/2}AD^{-1/2}$ (or $D^{-1/2}(D-A)D^{-1/2}$)	Normalized Laplacian	$[0, 2]$
$D^{-1/2}AD^{-1/2}$	Normalized adjacency matrix (Ng, Jordan, Weiss. 2001)	
$I - D^{-1}A$	Random Walk Laplacian	
$I + D^{-1/2}AD^{-1/2}$	1st order approximation GCN (Kipf and Welling. 2016)	
$\hat{A} = I + A$	Augmented adjacency matrix	
$\hat{D} - \hat{A} = (D+I) - (A+I) = D - A$	(Augmented) Laplacian	
$I - \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$	Normalized augmented Laplacian	
$\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$	Normalized augmented adjacency matrix GCN (Kipf and Welling. 2016)	$[-1, 1]$

These have similar eigenvectors (but differ in eigenvalues)

Goodness of adjacency matrices

- The use of adjacency matrix $\hat{S}_{\mathrm{adj}} = \theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2}$ allows GCN to be consider as spatial GNN (Gilmer *et al.* 2017)
  - Rewrite $\theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2} x$ as $\hat{A} H W$ it is clear that the method is spatial

- $\hat{S}_{\mathrm{adj}} = \theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2}$ as low-pass filter (Wu *et al.* 2019)
  - A filter $x * g = U g_\theta U^\top x$ projects x into the eigenbasis U
    - Adjacency matrices filters x through only the low frequency (global) eigenvectors
  - Two contributing factors
    1. Effects of stacking multiple layers
    2. Effects of augmentation

Goodness of adjacency matrices

- $\hat{S}_{\text{adj}} = \theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2}$ as low-pass filter (Wu *et al.* 2019)

1. Effects of stacking multiple layers
   - As mentioned, $\left(\hat{S}_{\text{adj}}\right)^k = \theta^k U (2-\Lambda)^k U^\top$
   - At high k, values of $(2-\Lambda)^k$ for $(2-\Lambda) \ll 1$ diminish

Filter	Eigenvalues (using the Karate club graph for L)
L^6	25.41, 17.54, 15.76, 14.95, 11.26, 9.24, 8.09, 7.31, 6.1, 4.16, 2.43, 1.82, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.56, 0.42, 0.31, 0.21, 0.16, 0.13, 0.07, 0.05, 0, 0, 0, 0
$(S_{\text{1-order}})^6$	64, 42.45, 25.26, 17.59, 7.14, 6.08, 4.67, 4., 3.45, 2.66, 2.14, 1.71, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.51, 0.35, 0.15, 0.07, 0.05, 0.04, 0.03, 0.02, 0.01, 0.01, 0, 0
$(\hat{S}_{\text{adj}})^6$	1, 0.52, 0.22, 0.12, 0.03, 0.02, 0.01, 0.01, 0.01, 0.01, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

  - Eigenvalue of 1 for $(\hat{S}_{\text{adj}})^6$ corresponds to the eigenvalue of 0 for $L \Rightarrow$ low frequency (low-pass) filter
  - The same cannot be achieved with L because of the range of eigenvalues

Goodness of adjacency matrices

- $\hat{S}_{\text{adj}} = \theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2}$ as low-pass filter (Wu *et al*. 2019)

  2. Effects of augmentation

     - An adjacency matrix with augmentation (self-loops) has a smaller spectrum than one without (that is, the normalized adjacency matrix $D^{-1/2}AD^{-1/2}$)
     - **Theorem** (Wu *et al*. 2019). Let A (and D) be the adjacency matrix (and degree matrix) of an undirected, weighted, simple connected graph G. Let $\hat{A} = A + \gamma I$, $\gamma > 0$ and let $\widehat{D}$ be its degree matrix. Let
       - λ_n/λ_1 be the min/max eigenvalues of $D^{-1/2}AD^{-1/2}$
       - $\hat{\lambda}_n/\hat{\lambda}_1$ be the min/max eigenvalues of $\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}$

       Then $\lambda_n < \hat{\lambda}_n < \hat{\lambda}_1 = \lambda_1 = 1$

     ⇒ Eigenvalues of $\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}$ range in $[\lambda, 1]$ for some λ > -1 ⇒ No exponential increase at large k

More levels of augmentation

- Extend augmentation $\hat{A} = A + I$ to more levels
  - Consider $\hat{A}_\gamma = A + \gamma I$ for different values of γ
    - The larger the value γ, the smaller the spectrum

      Theorem (Hoang and Maehara, 2019). Let
      - $\hat{A}_\gamma = A + \gamma I$ for $\gamma > 0$
      - $\widehat{D}_\gamma$ be the degree matrix for $\hat{A}_\gamma$
      - $\lambda_\gamma^{(i)}$ be the i^{th} largest eigenvalue of $\widehat{D}_\gamma^{-1/2} \hat{A}_\gamma \widehat{D}_\gamma^{-1/2}$

      Then for $0 \le \gamma' < \gamma$, $\lambda_{\gamma'}^{(i)} < \lambda_\gamma^{(i)} \le \lambda_{\gamma'}^{(1)} = \lambda_\gamma^{(1)} = 1$

      Corollary. $\gamma > \gamma' \Rightarrow [\lambda_\gamma^{(n)}, \lambda_\gamma^{(1)}]$ is smaller than $[\lambda_{\gamma'}^{(n)}, \lambda_{\gamma'}^{(1)}]$
    - Eigenvectors would change as well but that trend is less well understood

Eigenvalues after augmentation

- Eigenvalues of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ for the Karate club
  - All eigenvalues except 1 diminishes quickly when raised to some power
  - Negative eigenvalues will dovetail between negative and positive as the power changes between odd and even numbers
- **At some γ value, the range becomes close to [0, 1]**
  - In the present example, $\gamma = 4.5$

γ	**Eigenvalues of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$**	**Range**
0.0	1.0, 0.868, 0.713, ..., -0.583, -0.612, -0.715	[-0.715, 1.0]
0.5	1.0, 0.884, 0.747, ..., -0.391, -0.435, -0.542	[-0.542, 1.0]
1.0	**1.0, 0.896, 0.774, ..., -0.271, -0.312, -0.420**	**[-0.420, 1.0]**
1.5	1.0, 0.906, 0.796, ..., -0.182, -0.220, -0.325	[-0.325, 1.0]
2.0	1.0, 0.915, 0.815, ..., -0.113, -0.149, -0.249	[-0.249, 1.0]
2.5	1.0, 0.922, 0.830, ..., -0.057, -0.089, -0.184	[-0.184, 1.0]
3.0	1.0, 0.928, 0.843, ..., -0.010, -0.039, -0.129	[-0.129, 1.0]
3.5	1.0, 0.933, 0.854, ..., 0.032, 0.004, -0.080	[-0.080, 1.0]
4.0	1.0, 0.937, 0.864, ..., 0.069, 0.042, -0.037	[-0.037, 1.0]
4.5	**1.0, 0.941, 0.873, ..., 0.103, 0.076, 0.000**	**[0.000, 1.0]**
5.0	1.0, 0.945, 0.881, ..., 0.134, 0.106, 0.036	[0.036, 1.0]
5.5	1.0, 0.948, 0.888, ..., 0.163, 0.133, 0.067	[0.067, 1.0]
6.0	1.0, 0.950, 0.894, ..., 0.190, 0.158, 0.096	[0.096, 1.0]
6.5	1.0, 0.953, 0.899, ..., 0.215, 0.181, 0.123	[0.123, 1.0]
7.0	1.0, 0.955, 0.904, ..., 0.239, 0.203, 0.148	[0.147, 1.0]
7.5	1.0, 0.957, 0.909, ..., 0.261, 0.223, 0.170	[0.170, 1.0]
8.0	1.0, 0.959, 0.913, ..., 0.282, 0.242, 0.192	[0.192, 1.0]

Eigenvectors after augmentation

- Eigenvectors of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ for the Karate club
- Deviation from $D^{-1/2}AD^{-1/2}$ becomes very significant as γ increases beyond 1

γ	**Eigenvector of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ of largest eigenvalue (=1)**
0.0	0.320, 0.240, 0.253, 0.196, ..., 0.160, 0.196, 0.277, 0.330
0.5	0.309, 0.234, 0.246, 0.194, ..., 0.161, 0.194, 0.269, 0.318
1.0	**0.299, 0.229, 0.241, 0.192, ..., 0.162, 0.192, 0.262, 0.308**
1.5	0.291, 0.225, 0.236, 0.190, ..., 0.163, 0.190, 0.255, 0.299
2.0	0.283, 0.222, 0.231, 0.189, ..., 0.164, 0.189, 0.250, 0.291
2.5	0.277, 0.218, 0.228, 0.188, ..., 0.164, 0.188, 0.245, 0.284
3.0	0.271, 0.216, 0.224, 0.187, ..., 0.165, 0.187, 0.241, 0.278
3.5	-0.266, -0.213, -0.222, -0.186, ..., -0.165, -0.186, -0.237, -0.273
4.0	-0.262, -0.211, -0.219, -0.185, ..., -0.166, -0.185, -0.234, -0.268
4.5	**-0.258, -0.209, -0.217, -0.184, ..., -0.166, -0.184, -0.231, -0.264**
5.0	-0.254, -0.207, -0.215, -0.184, ..., -0.166, -0.184, -0.228, -0.260
5.5	-0.250, -0.206, -0.213, -0.183, ..., -0.166, -0.183, -0.226, -0.256
6.0	-0.247, -0.204, -0.211, -0.183, ..., -0.167, -0.183, -0.224, -0.253
6.5	-0.244, -0.203, -0.209, -0.182, ..., -0.167, -0.182, -0.222, -0.250
7.0	-0.242, -0.202, -0.208, -0.182, ..., -0.167, -0.182, -0.220, -0.247
7.5	0.239, 0.200, 0.206, 0.181, ..., 0.167, 0.181, 0.218, 0.244
8.0	-0.237, -0.199, -0.205, -0.181, ..., -0.167, -0.181, -0.216, -0.242

Eigenvectors after augmentation

- Eigenvectors of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ for the Karate club
- Deviation from $D^{-1/2}AD^{-1/2}$ becomes very significant as γ increases beyond 1

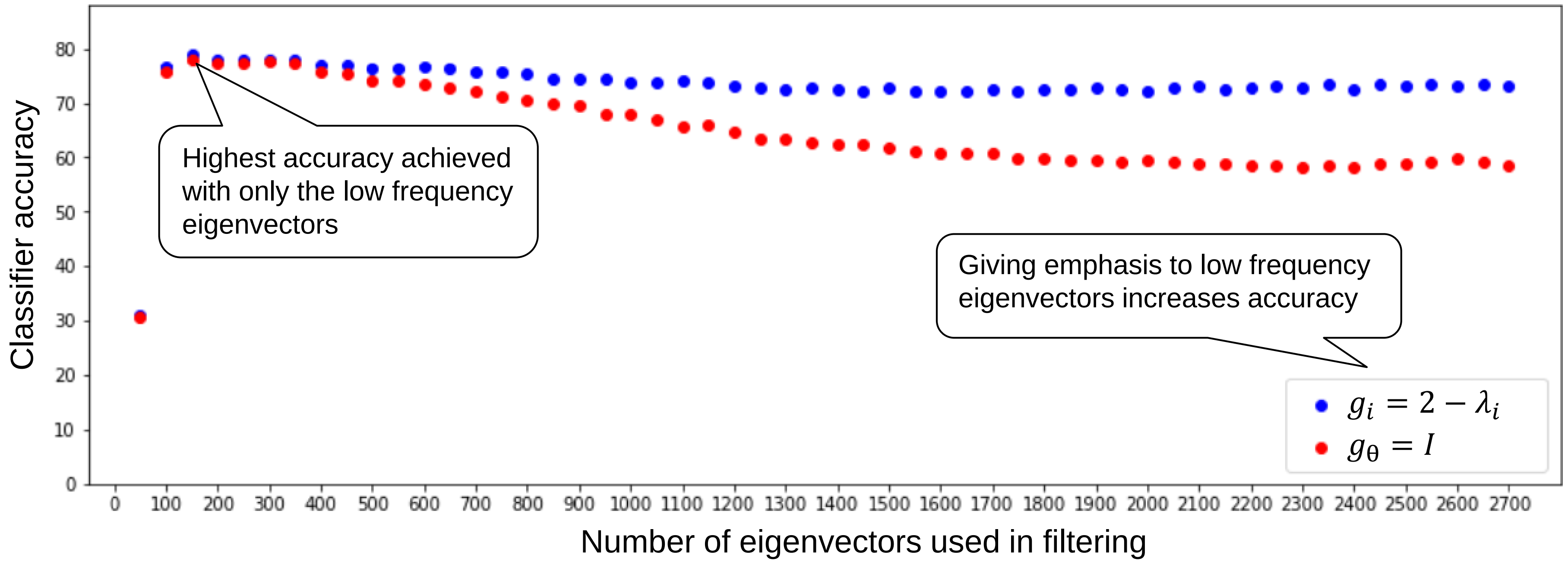
γ	**Eigenvector of $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ of smallest eigenvalue**
0.0	0.221, 0.185, 0.035, 0.019, ..., -0.164, -0.199, 0.410, 0.473
0.5	0.247, 0.198, 0.033, 0.015, ..., -0.169, -0.228, 0.410, 0.501
1.0	**0.273, 0.197, 0.031, 0.009, ..., -0.166, -0.242, 0.404, 0.524**
1.5	0.295, 0.190, 0.029, 0.004, ..., -0.161, -0.249, 0.396, 0.545
2.0	-0.315, -0.180, -0.028, 0.001, ..., 0.154, 0.251, -0.386, -0.566
2.5	0.333, 0.168, 0.027, -0.006, ..., -0.147, -0.250, 0.373, 0.585
3.0	0.349, 0.157, 0.027, -0.009, ..., -0.141, -0.247, 0.358, 0.604
3.5	0.364, 0.145, 0.027, -0.012, ..., -0.135, -0.243, 0.342, 0.621
4.0	0.377, 0.134, 0.027, -0.015, ..., -0.129, -0.239, 0.326, 0.637
4.5	**-0.388, -0.124, -0.028, 0.017, ..., 0.124, 0.234, -0.308, -0.653**
5.0	-0.398, -0.114, -0.029, 0.019, ..., 0.119, 0.229, -0.291, -0.667
5.5	-0.406, -0.105, -0.030, 0.020, ..., 0.115, 0.224, -0.274, -0.680
6.0	-0.413, -0.097, -0.031, 0.021, ..., 0.110, 0.219, -0.257, -0.692
6.5	-0.418, -0.089, -0.031, 0.022, ..., 0.107, 0.214, -0.241, -0.703
7.0	-0.422, -0.082, -0.032, 0.023, ..., 0.104, 0.209, -0.226, -0.714
7.5	0.425, 0.076, 0.033, -0.024, ..., -0.101, -0.204, 0.211, 0.723
8.0	-0.427, -0.070, -0.034, 0.024, ..., 0.098, 0.200, -0.197, -0.733

Low-pass filter performance

- We need to first find out how well a low-pass filter perform
- Construct such a filter (of only low frequency eigenvectors)
  - Recall that $x * g = U g_\theta U^\top x$ where $U = [\mu_1, \mu_2, \dots]$
    - $g_\theta = \mathrm{diag}(U^\top g) = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix}$
    - Each g_i weights the significance of eigenvector μ_i
  - Obtain U from decomposition of normalized Laplacian
    - The eigenvalues are in the range of $[0,2]$ where 0 has the lowest frequency (global) and 2 has the highest frequency
    - Sort eigenvectors by the eigenvalues and include only low frequency eigenvectors in filter $UIU^\top$ (details in later slide)
      - The use of I as g_θ implies that all eigenvectors included are equal
      - Alternatively let $g_i = 2 - \lambda_i$ so smaller eigenvalues are more significant
    - Compare effects of including only low frequency eigenvectors versus using all eigenvectors

Low-pass filter performance

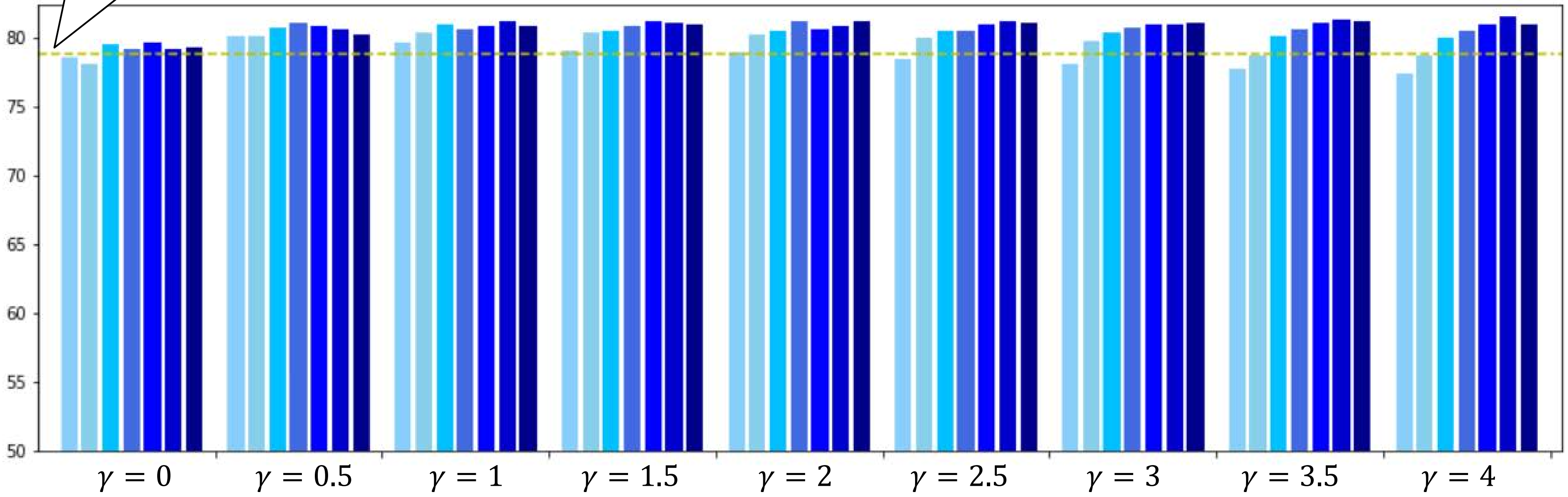
- Cora dataset
  - Nodes: 2708 scientific publications
  - Links: 5429
  - Feature: 1433 word embedding
  - Classes: 7
- Procedure
  - Filter features with 50, 100, 150, … eigenvectors of the lowest frequencies
  - Train a 2-layer MLP to classify with the filtered features



Adjacency matrix performance

- Repeat test with $\widehat{D}_\gamma^{-1/2}\hat{A}_\gamma\widehat{D}_\gamma^{-1/2}$ where $\hat{A}_\gamma = A + \gamma I$ as filter

Best performance of low-pass filter



- Accuracies obtained comparable to low-pass filters
- Increasing amount of augmentation γ improves accuracy
- Stacking more layers helps but only to a certain extend

Filter with only subset eigenvectors

- Recall from earlier slide

$$U^\top x = \begin{bmatrix} \leftarrow & \mu_1 & \rightarrow \\ \leftarrow & \mu_2 & \rightarrow \\ & \vdots & \end{bmatrix} x = \begin{bmatrix} \mu_1^\top x \\ \mu_2^\top x \\ \vdots \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix}$$

- Examine the exact form of a_i
  - Denote $x = \begin{bmatrix} \leftarrow & x_1 & \rightarrow \\ \leftarrow & x_2 & \rightarrow \\ & \vdots & \end{bmatrix}$, where each $x_i = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{iM}]$

$$U^\top x = \begin{bmatrix} \leftarrow & \mu_1 & \rightarrow \\ \leftarrow & \mu_2 & \rightarrow \\ & \vdots & \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ & \vdots & & \end{bmatrix} = \begin{bmatrix} \mu_1^\top x_{*1} & \mu_1^\top x_{*2} & \cdots & \mu_1^\top x_{*M} \\ \mu_2^\top x_{*1} & \mu_2^\top x_{*2} & \cdots & \mu_2^\top x_{*M} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$\Rightarrow a_i = [\mu_i^\top x_{*1} \quad \mu_i^\top x_{*2} \quad \cdots \quad \mu_i^\top x_{*M}]$$

  - a_i is computed from all rows and columns of x
  - For $\mu_i^\top x_{**}$ to compute correctly indices of $\mu_i^\top$ and x_{**} must match
  - However, the **ordering of $\mu_1, \mu_2, \ldots$ in $\begin{bmatrix} \leftarrow & \mu_1 & \rightarrow \\ \leftarrow & \mu_2 & \rightarrow \\ & \vdots & \end{bmatrix}$ does not matter**
    - To use only some eigenvectors, simply **zero out the unused eigenvectors** (corresponding a_is will become zero)
    - Or just **remove those unused eigenvectors**