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

| RecGNN |
| Graph Fourier Transform |
| Spectral ConvGNN |
| Spatial ConvGNN |

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

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

GNN history (significant eras)

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

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

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)

GoogLeNet and VGG

- Spectral domain filters as NNs and their approximation techniques

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

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

RecGNN

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

- 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

© 2021. Ng Yen Kaow

GNN history (people behind these)

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

| RecGNN |
| Graph Fourier Transform |
| Spectral ConvGNN |
| Spatial ConvGNN |

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

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

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

  x and H will be used interchangeably here

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

  where $a_i = \mu_i x$ is the projection onto μ_i

  - The projected space is $\sum_i a_i \mu_i$

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
$$x * g = F^{-1}\big(F(x) \odot F(g)\big) = U(U^{\mathsf{T}}x \odot U^{\mathsf{T}}g)$$
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 $U^{\mathsf{T}}g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \end{bmatrix}$. Let $g_\theta = \mathrm{diag}(U^{\mathsf{T}}g) = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix}$

Then we can write $x * g = U g_\theta U^{\mathsf{T}} x$ (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

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

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

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

- Hence 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 $O(N^2) \Rightarrow$ expensive

- Approximate g_θ with Chebyshev polynomials

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

where

- $\tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - I_N$ ($\lambda_{\max}$ 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)
  - $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$

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

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

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

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

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

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

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

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

where K is the number of expansion terms. 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 $T(L) = U T(\Lambda) U^\top$

$$x * g_{\theta'} \approx \sum_{i=0}^{K} \theta_i' T_i(\tilde{L}) x$$

© 2021. Ng Yen Kaow

Chebyshev approximation for U

☐ Hence we have $x * g_{\theta'} \approx \sum_{i=0}^{K} \theta_i' T_i(\tilde{L}) x$

☐ Furthermore, from the Chebyshev recurrence

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

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

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

☐ Then, $x * g_{\theta'} \approx \sum_{i=0}^{K} \theta_i' T_i(\tilde{L}) x = [\theta_0' \quad ... \quad \theta_K'] \begin{bmatrix} \bar{x}_0 \\ \vdots \\ \bar{x}_K \end{bmatrix}$

■ …and 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 $\theta_0', \ldots, \theta_K'$

$K = 1$ approximations (GCN)

- Hence we have $x * g_{\theta'} \approx \sum_{i=0}^{K} \theta_i' T_i(\tilde{L}) x$

- Finally, GCN takes $K = 1$ to obtain

$$x * g_{\theta'} \approx \theta_0' x + \theta_1' \tilde{L} x = \theta_0' x + \theta_1' \left(\frac{2}{\lambda_{\max}} L - I_N \right) x$$

  - Furthermore let $\lambda_{\max} = 2 \Rightarrow x * g_{\theta'} \approx \theta_0' x + \theta_1' (L - I_N) x$
  - Let θ_0' and θ_1' be the parameters to be learned

- Using the unweighted normalized Laplacian, $L = D^{-1/2}(D - A)D^{-1/2} = I_N - D^{-1/2}AD^{-1/2}$, then

$$x * g_{\theta'} = \theta_0' x - \theta_1' D^{-1/2}AD^{-1/2} x$$

- Further constraint the number of parameters by letting $\theta_0' = -\theta_1' = \theta$, $\boxed{x * g_{\theta'} = \theta \left(I_N + D^{-1/2}AD^{-1/2} \right) x}$

Kipf and Welling. Semi-Supervised Classification with Graph Convolutional Networks, 2016

$K = 1$ approximations (GCN)

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

$$\Rightarrow x * g_{\theta'} = \theta\left(I_N + D^{-1/2}AD^{-1/2}\right)x = \theta(2I_N - L)x$$

- Then, multiple applications of $\theta(2I_N - L)$ would result in

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

  where Λ/U are the eigenvalues/eigenvectors for L

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

  - 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 $\hat{A} = A + I$ and normalize $\hat{A}$ (renormalization)

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

  - How does this affect the spectral coefficients?

$K = 1$ approximations (GCN)

- ☐ Compare $\widehat{D}^{-1/2} \widehat{A} \widehat{D}^{-1/2}$ to $D^{-1/2} A D^{-1/2}$ (Ng, Weiss, and Jordan 2001)
    - ▪ Eigenvalues of $D^{-1/2} A D^{-1/2}$ range in [-1,1]
    - ▪ $\widehat{D}^{-1/2} \widehat{A} \widehat{D}^{-1/2}$ differs from $D^{-1/2} A D^{-1/2}$ in its self-loops ($\widehat{A} = A + I$)

- ☐ A Laplacian with self-loops has a smaller spectrum than one without
    - ▪ **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 $\widehat{A} = A + \gamma I$, $\gamma > 0$ and let $\widehat{D}$ be its degree matrix. Let
        - ☐ λ_1 / λ_n be the min/max eigenvalues of $D^{-1/2} A D^{-1/2}$
        - ☐ $\hat{\lambda}_1 / \hat{\lambda}_n$ be the min/max eigenvalues of $\widehat{D}^{-1/2} \widehat{A} \widehat{D}^{-1/2}$

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

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

How legit are GCN approximations

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

  1. $S_{\text{1-order}} = \theta\left(I_N + D^{-1/2}AD^{-1/2}\right)$, or

  2. $\hat{S}_{\text{adj}} = \theta\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ $(\hat{A} = A + I)$

  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,

  - $\theta'_0 = -\theta'_1 \Rightarrow x * g_{\theta'} = \theta(2I_N - L)x$
    $\Rightarrow$ 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

GCN properties

- ☐ GCN as spatial GNN (Gilmer *et al.* 2017)
  - ■ Consider $\hat{S}_{\text{adj}} = \theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2}$ ($\hat{A} = A + I$)
  - ■ Rewrite $\theta \widehat{D}^{-1/2} \hat{A} \widehat{D}^{-1/2} x$ as $\hat{A}HW$ and we arrive at a spatial method

- ☐ GCN as low-pass filter (Wu *et al.* 2019)
  - ■ As mentioned, $\hat{S}_{\text{adj}} = \theta^k U(2 - \Lambda)^k U^\top$
  - ■ At high k, values of $(2 - \Lambda)^k$ for $(2 - \Lambda) \ll 1$ diminishes
    - ☐ This often eliminates the negative $(2 - \Lambda)^k$ values (for odd k)

Filter	Eigenvalues
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

With the Karate club graph for L