

Graph Convolutional Network

Heting Gao

University of Illinois at Urbana-Champaign

hgao17@illinois.edu

November 10, 2018

1 Introduction

- About Graph

2 Graph Convolution

- Preliminary
- Graph Fourier Transform

3 (ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- A connected undirected graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$
 - \mathcal{V} is the set of vertices and $|\mathcal{V}| = N$.
 - \mathcal{E} is the set of edges.
 - \mathbf{W} is the weighted adjacency matrix.
 - $W_{i,j}$ is the weight of the edge $e = (i, j)$ connecting vertex i and j .
 - $W_{i,j} = 0$ if the edge does not exist.
 - If the weight of the graph is not naturally defined, a common way to define the weight is

$$W_{i,j} = \begin{cases} \exp\left(-\frac{[dist(i,j)]^2}{2\theta}\right) & \text{if } dist(i,j) \leq \kappa \\ 0 & \text{otherwise} \end{cases}$$

for some parameter κ and θ . $dist(i,j)$ can be the actual distance on the graph between vertex i and j , or the distance between features of vertex i and j

Preliminary

- A signal or a function on the graph $f : \mathcal{V} \rightarrow \mathbb{R}$ can be represented as a vector $\mathbf{f} \in \mathbb{R}^N$. $\mathbf{f}_i = f(v_i)$ is the function value on vertex $v_i \in \mathcal{V}$
- The Non-Normalized Graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$
 - \mathbf{W} is the weight matrix.
 - \mathbf{D} is the degree matrix. It is a diagonal matrix and diagonal element is the sum of all the incident edge weights.
 - $\mathbf{D}_{i,i} = \sum_{j=1}^N \mathbf{W}_{i,j}$
- The Graph Laplacian \mathbf{L} is a difference operator. $\forall \mathbf{f} \in \mathbb{R}^N$,

$$(\mathbf{L}f)(i) = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{i,j} [f(i) - f(j)]$$

$$(\mathbf{L}f)_i = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{i,j} (\mathbf{f}_i - \mathbf{f}_j)$$

where \mathcal{N}_i denote the set of neighbor nodes of vertex i .

- 1-D Laplacian operator Δ

$$\begin{aligned}f'(t) &= \lim_{h \rightarrow 0} \frac{f(t+h) - f(t)}{h} \\ \Delta f(t) &= \frac{\partial^2}{\partial t^2} f(t) \\ &= \frac{\partial}{\partial t} f'(t) \\ &= \lim_{h \rightarrow 0} \frac{f'(t+h) - f'(t)}{h}\end{aligned}$$

- 1-D discrete Laplacian operator Δ

$$\begin{aligned}f'[n] &= f[n+1] - f[n] \\ \Delta f[n] &= f'[n+1] - f'[n] \\ &= (f[n+1] - f[n]) - (f[n] - f[n-1]) \\ &= f[n+1] + f[n-1] - 2f[n]\end{aligned}$$

- 2-D discrete Laplacian operator Δ

$$\begin{aligned}\Delta f[n, m] &= f[n+1, m] + f[n-1, m] + f[n, m+1] + f[n, m-1] \\ &\quad - 4f[n, m]\end{aligned}$$

- The Graph Laplacian \mathbf{L} is a discrete Laplacian operator on the graph signals.

$$\begin{aligned}(\mathbf{L}\mathbf{f})_i &= \sum_{j \in \mathcal{N}_i} \mathbf{w}_{i,j}(\mathbf{f}_i - \mathbf{f}_j) \\ -\Delta \mathbf{f} &= \mathbf{L}\mathbf{f}\end{aligned}$$

Fourier Transform

- For a given function $f(t)$, its Fourier transform F at a given frequency $2\pi k$ is

$$F(\Omega) = \langle f(t), e^{j\Omega t} \rangle = \int_{\mathbb{R}} f(t) e^{-j\Omega t} dt$$

- The Laplacian of the basis $e^{j\Omega t}$ is in form of itself.

$$-\Delta e^{j\Omega t} = -\frac{\partial^2}{\partial t^2} e^{j\Omega t} = \Omega^2 e^{j\Omega t}$$

- For graph Fourier transform, we also want to find a set of analogous basis. Let $\mathbf{u} \in \mathbb{R}^N$ be a basis for graph transform, we want

$$-\Delta \mathbf{u} = \mathbf{L} \mathbf{u} = \lambda \mathbf{u}$$

- This is eigenvalue decomposition

Graph Fourier Transform

- Let $\mathbf{U} = [\mathbf{u}_l]_{l=1,\dots,N}$ denote the matrix of eigenvectors of \mathbf{L}
- Let $\mathbf{\Lambda} = [\lambda_l]_{l=1,\dots,N}$ denote the diagonal matrix of eigenvalues of \mathbf{L}
- For a given signal \mathbf{f} , its Fourier transform $\mathbf{F}(\lambda_l)$ at the given frequency λ_l is

$$\mathbf{F}(\lambda_l) = \langle \mathbf{f}, \mathbf{u}_l \rangle = \sum_{i=1}^N \mathbf{f}_i \mathbf{u}_{l,i}^*$$

- The inverse Fourier transform is then

$$\mathbf{f}_i = \sum_{l=1}^N \mathbf{F}(\lambda_l) \mathbf{u}_{l,i}$$

- Let $\mathbf{F} \in \mathbb{R}^N$ denote the Fourier transform vector of the given graph signal $\mathbf{f} \in \mathbb{R}^N$, we have the following matrix form of Fourier transform.

$$\begin{aligned}\mathbf{F} &= \mathbf{U}^T \mathbf{f} \\ \mathbf{f} &= \mathbf{U} \mathbf{F}\end{aligned}$$

Graph Spectral Filtering

- Let $\mathcal{F} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ denote the graph Fourier transform Let $\mathcal{F}^{-1} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ denote the inverse graph Fourier transform.
- Let $\mathbf{h} \in \mathbb{R}^N$ denote the filter function on the graph.
- Let $\mathbf{H} \in \mathbb{R}^N$ denote the Fourier transform of the filter function.
- Let $\mathbf{y} \in \mathbb{R}^N$ denote the function after filtering on the graph.

$$\begin{aligned}\mathbf{y} &= \mathbf{h} * \mathbf{f} \\ &= \mathcal{F}^{-1}[\mathcal{F}(\mathbf{h}) \odot \mathcal{F}(\mathbf{f})] \\ &= \mathbf{U}[\mathbf{U}^T \mathbf{h} \odot \mathbf{U}^T \mathbf{f}] \\ &= \mathbf{U}[\mathbf{H} \odot \mathbf{U}^T \mathbf{f}] \\ &= \mathbf{U} \begin{bmatrix} \mathbf{H}(\lambda_1) & & \\ & \ddots & \\ & & \mathbf{H}(\lambda_l) \end{bmatrix} \mathbf{U}^T \mathbf{f}\end{aligned}$$

(NIPS-2016) [Michal Defferrard] Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

- Define $\mathbf{H}(\mathbf{L})$ the spectral filter as

$$\mathbf{H}(\mathbf{L}) = \mathbf{U} \begin{bmatrix} \mathbf{H}(\lambda_1) & & \\ & \ddots & \\ & & \mathbf{H}(\lambda_l) \end{bmatrix} \mathbf{U}^T$$

- The adjustable parameter would be $[\mathbf{H}_l]_{l=1,2,\dots,N}$
- Let $\theta = [\mathbf{H}(\lambda_l)]_{l=1,2,\dots,N}$.
- Let $g_\theta(\mathbf{\Lambda}) = \text{diag}(\theta)$ We can define the convolutional layer as

$$\mathbf{y} = \sigma(\mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{f})$$

Graph Convolutional Neural Network

- If we define the convolutional layer as

$$\mathbf{y} = \sigma(\mathbf{U}g_{\theta}(\mathbf{\Lambda})\mathbf{U}^T\mathbf{f})$$

- There are however 3 limitations.
 - The convolution is not localized. With arbitrary θ , the signal \mathbf{f} can be propagated to any other nodes.
 - $\theta \in \mathbf{R}^N$ means that we need N parameter.
 - Eigen decomposition has a computational complexity of $\mathcal{O}(N^3)$ and every forward propagation has complexity of $\mathcal{O}(N^2)$

Graph Convolutional Neural Network

- We can instead define

$$g_{\theta}(\mathbf{\Lambda}) = \sum_{k=1}^K \theta_k \mathbf{\Lambda}^k$$

- We get the new convolutional layer as

$$\begin{aligned} \mathbf{y} &= \sigma(\mathbf{U} g_{\theta}(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{f}) \\ &= \sigma(\mathbf{U} (\sum_{k=1}^K \theta_k \mathbf{\Lambda}^k) \mathbf{U}^T \mathbf{f}) \\ &= \sigma(\sum_{k=1}^K \theta_k \mathbf{U} \mathbf{\Lambda}^k \mathbf{U}^T \mathbf{f}) \\ &= \sigma(\sum_{k=1}^K \theta_k \mathbf{L}^k \mathbf{f}) \end{aligned}$$

Graph Convolutional Neural Network

- The new definition of a convolutional layer is

$$\mathbf{y} = \sigma\left(\sum_{k=1}^K \theta_k \mathbf{L}^k \mathbf{f}\right)$$

- It has three advantages
 - The convolution is localized and is exactly K -hop localized we are using at most K 's power of \mathbf{L}
 - We need only K parameters
 - We do not need to decompose \mathbf{L} and the forward propagation can be approximated using Chebyshev polynomials (I do not understand this part but I will still try to describe the steps described in the paper).

Chebyshev Polynomial

- Chebyshev Polynomial Expansion

$$T_0(y) = 1$$

$$T_1(y) = y$$

$$T_k(y) = 2yT_{k-1} - T_{k-2}$$

- These polynomials forms an orthogonal basis for $x \in L^2([-1, 1], \frac{dy}{\sqrt{1-y^2}})$, the Hilbert space of square integrable functions with respect to the measure $\frac{dy}{\sqrt{1-y^2}}$

$$\int_{-1}^1 \frac{T_l(y) T_m(y)}{\sqrt{1-y^2}} dy = \begin{cases} \delta_{l,m} \pi / 2 & m, l > 0 \\ \pi & m = l = 0 \end{cases}$$

Chebyshev Polynomial

- In particular, $\forall h \in L^2([-1, 1], \frac{dy}{\sqrt{1-y^2}})$, h has the following chebyshev polynomial expansion.

$$h(y) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(y)$$

- Since $\lambda \in [0, \lambda_{max}]$ plug in $y = \mathbf{U}(\frac{2\mathbf{\Lambda}}{\lambda_{max}} - \mathbf{I})\mathbf{U}^T = \frac{2\mathbf{L}}{\lambda_{max}} - \mathbf{I}$,

$$g_{\theta}(\mathbf{L}) = \sum_{k=1}^K \theta_k \mathbf{L}^k = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(y)$$

where $T_k(y)$ can be computed recursively as

$$T_k(y) = 2yT_{k-1}(y) + T_{k-2}(y)$$

Chebyshev Polynomial

- Let $\bar{\mathbf{f}}_k = T_k(y)\mathbf{f}$ can be computed recursively as

$$\begin{aligned}\bar{\mathbf{f}}_k &= T_k(y)\mathbf{f} \\ T_k(y)\mathbf{f} &= 2yT_{k-1}(y)\mathbf{f} + T_{k-2}(y)\mathbf{f} \\ &= 2y\bar{\mathbf{f}}_{k-1} + \bar{\mathbf{f}}_{k-2} \\ &= 2\left(\frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}\right)\bar{\mathbf{f}}_{k-1} + \bar{\mathbf{f}}_{k-2}\end{aligned}$$

- The approximated convolutional layer is as

$$y = \sigma(g_{\theta}(\mathbf{L})\mathbf{f}) = \sigma\left(\sum_{k=0}^K \theta_k \bar{\mathbf{f}}_k\right)$$

with $\bar{\mathbf{f}}_0 = \mathbf{f}$, and $\bar{\mathbf{f}}_1 = y\mathbf{f} = \left(\frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}\right)\mathbf{f}$

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- Instead of using K -hop localized filter, set $K = 1$, but instead stack multiple layers.
- Use symmetric normalized Laplacian

$$\mathbf{L}^{sym} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$$

- The entries of \mathbf{L}^{sym} are

$$\mathbf{L}_{i,j}^{sym} = \begin{cases} 1 & , i = j \\ \frac{1}{\sqrt{d_i d_j}} & , i \neq j \text{ and vertex } i \text{ and } j \text{ are connected} \\ 0 & , \text{otherwise} \end{cases}$$

- The equation is equivalent to

$$(\mathbf{L}^{sym} \mathbf{f})_i = \frac{1}{\sqrt{d_i}} \sum_{j \in \mathcal{N}_i} \mathbf{W}_{i,j} \left(\frac{\mathbf{f}_j}{\sqrt{d_i}} - \frac{\mathbf{f}_i}{\sqrt{d_j}} \right)$$

- The eigenvalues $[\lambda_l]_{l=1,2,\dots,N}$ of \mathbf{L}^{sym} is in the range of $[0, 2]$

ⁱKipf's paper uses \mathbf{A} to represent the weight matrix. I will stick to \mathbf{W} to be consistent in this presentation

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- Then the convolutional layer can be approximated as

$$\begin{aligned}y &= \sigma(g_{\theta}(\mathbf{L})\mathbf{f}) \\&\approx \sigma(\theta_0\mathbf{f} + \theta_1\mathbf{y}\mathbf{f}) \\&= \sigma(\theta_0\mathbf{f} + \theta_1(\frac{2\mathbf{L}^{sym}}{\lambda_{max}} - \mathbf{I})\mathbf{f}) \\&= \sigma[(\theta_0 - \theta_1)\mathbf{f} + \theta_1\mathbf{L}^{sym}\mathbf{f}] \text{ (assume } \lambda_{max} = 2) \\&= \sigma[(\theta_0 - \theta_1)\mathbf{f} + \theta_1(\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{f}] \\&= \sigma(\theta_0\mathbf{f} - \theta_1\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\mathbf{f})\end{aligned}$$

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- The approximated output layer

$$y = \sigma(\theta_0 \mathbf{f} - \theta_1 \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}} \mathbf{f})$$

- The number of parameters is further reduced to 1 in the paper by assuming $\theta = \theta_0 = -\theta_1$

$$y = \sigma(\theta(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}) \mathbf{f})$$

- The matrix $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ has eigenvalues $\lambda \in [0, 2]$. Repeated application on this matrix can result in numerical instability.
- Renormalize $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ to $\widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{W}} \widetilde{\mathbf{D}}^{-\frac{1}{2}}$

$$\begin{aligned}\widetilde{\mathbf{W}} &= \mathbf{W} + \mathbf{I} \\ \widetilde{\mathbf{D}}_{i,i} &= \sum_j \widetilde{\mathbf{W}}_{i,j}\end{aligned}$$

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- The final version of graph convolutional network is

$$\mathbf{Z} = \widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{W}} \widetilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{F} \Theta$$

where

- $\mathbf{F} \in \mathbb{R}^{N \times C}$ is the signal matrix.
- $\Theta \in \mathbb{R}^{C \times F}$ is the learnable parameters of the filter.
- $\mathbf{Z} \in \mathbb{R}^{N \times F}$ is the output matrix.
- C is the number of input feature channels.
- F is the number of output feature channels.

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- The paper performed semi-supervised node classification using the following GCN architecture

$$\mathbf{Z} = f(\mathbf{F}, \mathbf{W}) = \text{softmax}(\hat{\mathbf{W}} \text{ReLU}(\hat{\mathbf{W}} \mathbf{F} \Theta^{(0)}) \Theta^{(1)})$$

where $\hat{\mathbf{W}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{W}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$

- The loss function \mathcal{L} is defined on all the labeled nodes

$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F \mathbf{Y}_{lf} \ln \mathbf{Z}_{lf}$$

where \mathcal{Y}_L is the set of labeled node indices, \mathbf{Y} is the set of true labels.

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- The experiments are performed on the following dataset

Table 1: Dataset statistics, as reported in Yang et al. (2016).

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

- The results are

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	70.3 (7s)	81.5 (4s)	79.0 (38s)	66.0 (48s)
GCN (rand. splits)	67.9 \pm 0.5	80.1 \pm 0.5	78.9 \pm 0.7	58.4 \pm 1.7

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- Model comparison

Table 3: Comparison of propagation models.

Description	Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$ $K = 2$ $\sum_{k=0}^K T_k(\tilde{L})X\Theta_k$	69.8 69.6	79.5 81.2	74.4 73.8
1 st -order model (Eq. 6)	$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)	$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)	$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	70.3	81.5	79.0
1 st -order term only	$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron	$X\Theta$	46.5	55.1	71.4

(ICLR-2017) [Thomas N. Kipf] Convolutional Graph Network

- The limitation of this paper
 - Memory requirement is high. Each iteration requires the entire adjacency matrix. Does not work for large dense graphs.
 - Directed edges and edge features cannot be naturally Incorporated into this model.
 - The adding of \mathbf{I} to adjacency matrix \mathbf{W} is assuming the equal importance of self-connection and edges to neighbor nodes. It might be useful to introduce a weight λ on self-loop

$$\widetilde{\mathbf{W}} = \mathbf{W} + \lambda \mathbf{I}$$

- The convolutional layer only captures 1-hop locality. The expressiveness of the filter is still limited.

- (IEEE-2012) [David i Shuman] The Emerging Field of Signal Processing on Graphs
- (NIPS-2016) [Michal Defferrard] Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering
- (ICLR-2017) [Thomas N. Kipf] Semi-Supervised Classification with Graph Convolutional Networks
- This presentation follows the outline of this post
<https://www.zhihu.com/question/54504471>

The End