A brief introduction from Spectral CNN to SGC

Jie Zhang

Outline

- Spectral CNN
- ChebyNet
- GCN
- SGC
- Over-smooth issue for l-layers low-pass filter GNNs

Why using GNN?

- Euclidean data
  - Audio signals
  - Images
- Non-Euclidean data
  - Social networks
  - Point Cloud

Convolution on Graphs

- Spectral methods: define convolution in the spectral domain
    - Convolution is defined via graph Fourier transform and convolution theorem
    - The main challenge is that convolution filter defined in spectral domain is not localized in spatial/vertex domain
    - E.g., Spectral CNN, ChebyNet, and GCN
- Spatial methods: define convolution in the spatial/vertex domain
    - Convolution is defined as a weighted average function over all vertices located in the neighborhood of target vertex
    - The main challenge is that the size of neighborhood varies remarkably across nodes, e.g., power-law degree distribution
    - E.g., GraphSAGE, and GAT

GNNs in early phase

- Spectral Convolution Neural Networks (2014)
- ChebyNet (2016)
- Graph Convolution Networks (2017)
- Simple Graph Convolutional Networks (2019)

Define Graph

- Represent the weighted undirected graph as $G = (V, E)$,
  - where V is a set of vertices $|V| = n$,
  - E is a set of edges,
  - and A is a binary or weighted adjacency matrix representing the vertices proximity.

Weighted Adjacency Matrix

- Compute the weighted adjacency matrix using thresholded Gaussian kernel as
  - $$a_{ij} = \begin{cases} exp(-\frac{[dist(v_i, v_j)]^2}{\sigma^2}), & if\ dist(v_i, v_j) \leq k \\ 0, & otherwise \end{cases},$$
  - where a_{ij} represents the edge weight between vertices v_i and v_j,
  - $dist(v_i, v_j)$ denotes the physical distance between vertices v_i and v_j, or the Euclidean distance between two feature vectors describing v_i and v_i,
  - σ is the standard deviation of distances and k is the threshold.

Define Convolution

- Let f and g be two functions with convolution $f * g$.
- Let F denotes the Fourier transform operator, then $F\{f\}$ and $F\{g\}$ are the Fourier transform of f and g, respectively.
- And by applying the inverse Fourier transform F^{-1}, then $f * g = F^{-1}\{F\{f\} * F\{g\}\}$.

Laplacian matrix

- Diagonal degree matrix $D_{row\ ii} = \sum_j A_{ij}$ and $D_{column\ ii} = \sum_i A_{ij}$
- Combinatorial Laplacian: $L_{com} = D - A \in R^{n \times n}$
- Symmetric Normalized Laplacian: $L_{sym} = D^{-\frac{1}{2}} L_{com} D^{-\frac{1}{2}} = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
- Random Walk Normalized Laplacian: $L_{rw} = D^{-1} L_{com} = I_n - D_{row}^{-1} A = I_n - A D_{column}^{-1}$

- $L = U\Lambda U^{-1} = U\Lambda U^{\mathrm{T}}$,
    - where $\Lambda = diag([\lambda_0, \ldots, \lambda_{n-1}]) \in R^{n\times n}$,
    - $U = [u_0, \ldots, u_{n-1}] \in R^{n\times n}$ is the Fourier basis

Graph Fourier Transform and Graph Fourier Inverse Transform

- The graph Fourier transform of a signal x is defined as
  - $\hat{x} = U^{\mathrm{T}} x$
- The graph Fourier inverse transform is defined as
  - $x = U\hat{x}$

Define convolution in spectral domain

- Given a signal x as input and the other signal y as a filter, graph convolution $\star_G$ could be defined as
  - $x \star_G y = U((U^{\mathrm{T}}x) \odot (U^{\mathrm{T}}y))$

- A signal x is filtered by g_θ as
  - $g_\theta(L)x = g_\theta(U\Lambda U^{\mathrm{T}})x = Ug_\theta(\Lambda)U^{\mathrm{T}}x$
- A non-parametric filter, i.e. a filter whose parameters are all free, would be defined as
  - $g_\theta(\Lambda) = diag(\theta)$
  - where the parameter $\theta \in R^n$ is a vector of Fourier coefficients

Shortcomings of Spectral CNN

- Requiring eigen-decomposition of Laplacian matrix
  - Eigenvectors are explicitly used in convolution
- High computational cost
  - Multiplication with graph Fourier basis U is $O(n^2)$
- Not localized in spatial domain

Polynomial parametrization for localized filters

- Let $g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k \Lambda^k$, then
- $g_\theta(L)x \approx U \sum_{k=0}^{K-1} \theta_k \Lambda^k U^{\mathrm{T}} x = \sum_{k=0}^{K-1} \theta_k U \Lambda^k U^{\mathrm{T}} x = \sum_{k=0}^{K-1} \theta_k L^k x$
- Time complexity is still $O(n^2)$ because of the multiplication with the graph Fourier basis U.

Chebyshev polynomials of the first kind

- The Chebyshev polynomials of the first kind is defined as
  - $T_k(x) = cos(k \cdot arccos(x))$,
  - and recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, where
    - $T_0(x) = 1$
    - $T_1(x) = x$

- $g_\theta(\Lambda)$ can be well-approximated by a truncated expansion in terms of order $K-1$:
  - $g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda})$
  - where the parameter $\theta \in R^K$ is a vector of Chebyshev coefficients
  - and $T_k(\widetilde{\Lambda}) \in R^{n \times n}$ is the Chebyshev polynomial of order k evaluated at $\widetilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n,$
  - a diagonal matrix of scaled eigenvalues that lie in $[-1,\ 1]$ because of the $arccos(\cdot)$ function.

- $$g_\theta(L)x \approx U \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda}) U^{\mathrm{T}} x$$

- $$g_\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(U \widetilde{\Lambda} U^{\mathrm{T}}) x$$

- $$g_\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L}) x$$

Graph Convolution from ChebyNet

- The filtering operation can then be written as
- $g_\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})x$
- with scaled Laplacian $\widetilde{L} = \frac{2L}{\lambda_{max}} - I_n$, where $L = L_{sym}$ or $L = L_{rw}$

Recurrence relation

- $\bar{x}_k = 2\widetilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$, where $\bar{x}_0 = x$, and $\bar{x}_1 = \widetilde{L}x$
- $g_\theta(L)x \approx [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta$

Strengths of Chebyshev Polynomials Approximation

- Eigen-decomposition is not required
- Computational cost is $O(K|E|) \ll O(n^2)$

Graph Convolution from GCN

- In this linear formulation of a GCN, we further approximate $\lambda_{max} \approx 2$, and using L_{sym}.
- $g_\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})x$ simplifies to
    - $g_\theta(L)x \approx \theta_0 x + \theta_1(L - I_n)x = \theta_0 x - \theta_1(D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x$
    - with two free parameters θ_0 and θ_1.
- The filter parameters can be shared over the whole graph.

Graph Convolution from GCN

- Let $\theta = \theta_0 = -\theta_1$, then $g_\theta(L)x \approx \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x$.
- To alleviate the gradient exploding or vanishing issue, we introduce the renormalization trick:
- $I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$, with $\widetilde{A} = A + I_n$, and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$
- Finally, $\hat{L} = (D + I_n)^{-\frac{1}{2}}(A + I_n)(D + I_n)^{-\frac{1}{2}} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$, then $g_\theta(L)x \approx g_\theta(\hat{L})x = \theta\hat{L}x$

2-layers GCN

- $H^{(0)} = \phi(\hat{L}X\Theta^{(0)})$, where $\phi(\cdot)$ is the activation function like $ReLU$.
- $H^{(1)} = \hat{L}H^{(0)}\Theta^{(1)}$
- $\hat{Y}_{GCN} = softmax(H^{(1)})$

l-layers GCN

- $H^{(0)} = \phi(\hat{L}X\Theta^{(0)})$, where $\phi(\,\cdot\,)$ is the activation function like $ReLU$.
- $H^{(l-1)} = \hat{L}H^{(l-2)}\Theta^{(l-1)}$
- $\hat{Y}_{GCN} = softmax(H^{(l-1)})$

SGC

- $\hat{Y}_{SGC} = softmax(\hat{L}^l X \Theta)$, where $\hat{L}^l = \hat{L}\hat{L}\ldots\hat{L}$

Over-smoothing for l-layers GNNs

Causes:

1. Vanishing gradient
2. Low-pass or high-pass filter
3. more?