LanczosNet: Multi-Scale Deep Graph Convolutional Networks

Credit: Renjie Liao, Zhizhen Zhao, Raquel Urtasun, Richard S. Zemely

https://qdata.github.io/deep2Read

Presenter: Ryan McCampbell
https://qdata.github.io/deep2Read

Outline

Outline

Introduction

Two main issues with current GCN approaches

1. How to efficiently leverage multi-scale information
   - Graph coarsening - fixed process
   - Powers of graph Laplacian - expensive
2. Spectral filters are mostly fixed
   - Learning filters can produce more useful representations

Idea:

- Use low-rank approximation of graph Laplacian
  - Enables efficient computation of matrix powers for multi-scale information
- Design learnable spectral filters

Outline

Graph Fourier Transform

- Graph Laplacian $L = D - A$, $L = I - D^{-1}A$, or $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$
- Affinity matrix $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$
- Spectral decomposition $S = U\Lambda U^T$
- Graph Fourier Transform $Y = U^T X$ and $\hat{X} = UY$
- We can filter in the spectral domain

Localized Polynomial Filter

- τ-localized polynomial filter:

$$g_w(\Lambda) = \sum_{t=0}^{\tau-1} w_t \Lambda^t$$

  - Leverages information from nodes $\leq \tau$ hops away
- More general form:

$$Y = \sum_{t=0}^{\tau-1} g_t(S, ..., S^t, X) W_t$$

- Krylov subspace $\mathcal{K}_t(S, x) = span\{x, Sx, ..., S^{t-1}x\}$

Outline

Lanczos Algorithm

- Given affinity matrix S and node features x, the N-step Lanczos algorith computes orthogonal Q and symmetric tridiagonal T with $Q^T SQ = T$.

$$T = \begin{bmatrix} \gamma_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{N-1} \\ & & \beta_{N-1} & \gamma_N \end{bmatrix}$$

- Q forms orthogonal basis of $\mathcal{K}_N(S, x)$
- First K cols of Q form orthogonal basis of $\mathcal{K}_K(S, x)$

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.

Localized polynomial filter

- Run Lanczos for K steps starting with X_i to compute orthonormal basis Q of $\mathcal{K}_K(S, X_i)$

$$Y_j = Qw_{i,j}$$

- Q depends on X_i: separate run of Lanczos is needed for each graph convolution layer
- Ideally, we want to only compute Lanczos once during inference on a graph

Spectral Filter

Alternate view:

- Choose random starting vector x
- Treat K step Lanczos output as low-rank approximation $S \approx QTQ^T$
- Decompose tridiagonal matrix into Ritz values $T = BRB^T$
  - R is diagonal: approximation of eigenvalues
- $S \approx VRV^T$ where $V = QB$
- Rewrite graph convolution as

$$Y_j = [X_i, SX_i, ..., S^{K-1}X_i]w_{i,j} \approx [X_i, VRV^TX_i, ..., VR^{K-1}V^TX_i]w_{i,j}$$

Learning the Spectral Filter

- Use K different spectral filters with kth output $\hat{R}(k) = f_k([R^1, ..., R^{K-1}])$, where f_k is MLP

$$Y_j = [X_i, V\hat{R}(1)V^T X_i, ..., V\hat{R}(K-1)V^T X_i]w_{i,j}$$

Multi-Scale Graph Convolution

$$Y = [L^{\mathcal{S}_1}X, ..., L^{\mathcal{S}_M}X, V\hat{R}(\mathcal{I}_1)V^TX, ..., V\hat{R}(\mathcal{I}_N)V^TX]W$$

- $\mathcal{S}$: Short-scale parameters, e.g. $0, ..., 5$
- $\mathcal{I}$: Long-scale parameters, e.g. $10, 20, ..., 50$

LanczosNet

Algorithm 2 : LanczosNet

Input: Signal X, Lanczos output V and R, scale index sets $\mathcal{S}$ and $\mathcal{I}$,
Initialization: $Y_0 = X$
For $\ell = 1, 2, \dots, \ell_c$:
    $Z = Y_{\ell-1}$, $\mathcal{Z} = \{\emptyset\}$
    For $j = 1, 2, \dots, \max(\mathcal{S})$:
        $Z = SZ$
        If $j \in \mathcal{S}$:
            $\mathcal{Z} = \mathcal{Z} \cup Z$
    For $i \in \mathcal{I}$:
        $\mathcal{Z} = \mathcal{Z} \cup V\hat{R}(\mathcal{I}_i)V^\top Y_{\ell-1}$
    $Y_\ell = \mathrm{concat}(\mathcal{Z})W_\ell$
    If $\ell < L$
        $Y_\ell = \mathrm{Dropout}(\sigma(Y_\ell))$
Return Y_{ℓ_c}.

Outline

AdaLanczosNet

- Back-propagate through Lanczos algorithm
- Facilitates learning graph kernel and/or node embeddings

Graph Kernel

- Learnable anisotrophic graph kernel

$$k(x_i, x_j) = \exp\left(\frac{-\|f_\theta(x_i) - f_\theta(x_j)\|^2}{\epsilon}\right)$$

- f_θ is an MLP
- We can construct adjacency matrix $A_{i,j} = k(x_i, x_j)$ if $(i, j) \in \mathcal{E}$ and 0 otherwise
- Use to define affinity matrix $S = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
- We can discard f to learn node embeddings on X

Tridiagonal Decomposition

- Backpropagation through the eigendecomposition of tridiagonal matrix is unstable
- Instead directly use approximation $S = QTQ^T$

$$Y = [L^{\mathcal{S}_1}X, ..., L^{\mathcal{S}_M}X, Qf_1(T^{\mathcal{I}_1})Q^TX, ..., Qf_N(T^{\mathcal{I}_N})Q^TX]W$$

- f_i is learnable filter

Outline

Citation Networks

Cora	GCN-FP	GGNN	DCNN	ChebyNet	GCN	MPNN	GraphSAGE	GAT	LNet	AdaLNet
Public	74.6 ± 0.7	77.6 ± 1.7	79.7 ± 0.8	78.0 ± 1.2	80.5 ± 0.8	78.0 ± 1.1	74.5 ± 0.8	**82.6 ± 0.7**	79.5 ± 1.8	80.4 ± 1.1
3%	71.7 ± 2.4	73.1 ± 2.3	76.7 ± 2.5	62.1 ± 6.7	74.0 ± 2.8	72.0 ± 4.6	64.2 ± 4.0	56.8 ± 7.9	76.3 ± 2.3	**77.7 ± 2.4**
1%	59.6 ± 6.5	60.5 ± 7.1	66.4 ± 8.2	44.2 ± 5.6	61.0 ± 7.2	56.7 ± 5.9	49.0 ± 5.8	48.6 ± 8.0	66.1 ± 8.2	**67.5 ± 8.7**
0.5%	50.5 ± 6.0	48.2 ± 5.7	59.0 ± 10.7	33.9 ± 5.0	52.9 ± 7.4	46.5 ± 7.5	37.5 ± 5.4	41.4 ± 6.9	58.1 ± 8.2	**60.8 ± 9.0**
Citeseer	GCN-FP	GGNN	DCNN	ChebyNet	GCN	MPNN	GraphSAGE	GAT	LNet	AdaLNet
Public	61.5 ± 0.9	64.6 ± 1.3	69.4 ± 1.3	70.1 ± 0.8	68.1 ± 1.3	64.0 ± 1.9	67.2 ± 1.0	**72.2 ± 0.9**	66.2 ± 1.9	68.7 ± 1.0
1%	54.3 ± 4.4	56.0 ± 3.4	62.2 ± 2.5	59.4 ± 5.4	58.3 ± 4.0	54.3 ± 3.5	51.0 ± 5.7	46.5 ± 9.3	61.3 ± 3.9	**63.3 ± 1.8**
0.5%	43.9 ± 4.2	44.3 ± 3.8	53.1 ± 4.4	45.3 ± 6.6	47.7 ± 4.4	41.8 ± 5.0	33.8 ± 7.0	38.2 ± 7.1	53.2 ± 4.0	**53.8 ± 4.7**
0.3%	38.4 ± 5.8	36.5 ± 5.1	44.3 ± 5.1	39.3 ± 4.9	39.2 ± 6.3	36.0 ± 6.1	25.7 ± 6.1	30.9 ± 6.9	44.4 ± 4.5	**46.7 ± 5.6**
Pubmed	GCN-FP	GGNN	DCNN	ChebyNet	GCN	MPNN	GraphSAGE	GAT	LNet	AdaLNet
Public	76.0 ± 0.7	75.8 ± 0.9	76.8 ± 0.8	69.8 ± 1.1	77.8 ± 0.7	75.6 ± 1.0	76.8 ± 0.6	76.7 +- 0.5	**78.3 ± 0.3**	78.1 ± 0.4
0.1%	70.3 ± 4.7	70.4 ± 4.5	73.1 ± 4.7	55.2 ± 6.8	73.0 ± 5.5	67.3 ± 4.7	65.4 ± 6.2	59.6 +- 9.5	**73.4 ± 5.1**	72.8 ± 4.6
0.05%	63.2 ± 4.7	63.3 ± 4.0	66.7 ± 5.3	48.2 ± 7.4	64.6 ± 7.5	59.6 ± 4.0	53.0 ± 8.0	50.4 +- 9.7	**68.8 ± 5.6**	66.0 ± 4.5
0.03%	56.2 ± 7.7	55.8 ± 7.7	60.9 ± 8.2	45.3 ± 4.5	57.9 ± 8.1	53.9 ± 6.9	45.4 ± 5.5	50.9 +- 8.8	60.4 ± 8.6	**61.0 ± 8.7**

Table 1: Test accuracy with 10 runs on citation networks. The public splits in Cora, Citeseer and Pubmed contain 5.2%, 3.6% and 0.3% training examples respectively.

Methods	Validation MAE ($\times 1.0e^{-3}$)	Test MAE ($\times 1.0e^{-3}$)
GCN-FP [29]	15.06 ± 0.04	14.80 ± 0.09
GGNN [37]	12.94 ± 0.05	12.67 ± 0.22
DCNN [8]	10.14 ± 0.05	9.97 ± 0.09
ChebyNet [7]	10.24 ± 0.06	10.07 ± 0.09
GCN [11]	11.68 ± 0.09	11.41 ± 0.10
MPNN [62]	11.16 ± 0.13	11.08 ± 0.11
GraphSAGE [39]	13.19 ± 0.04	12.95 ± 0.11
GPNN [40]	12.81 ± 0.80	12.39 ± 0.77
GAT [33]	11.39 ± 0.09	11.02 ± 0.06
LanczosNet	$\mathbf{9.65 \pm 0.19}$	$\mathbf{9.58 \pm 0.14}$
AdaLanczosNet	10.10 ± 0.22	9.97 ± 0.20

Table 2: Mean absolute error on QM8 dataset.

Outline

Conclusions

- This method enables more powerful learning on graphs, incorporating multi-scale information as well as learned spectral filters and graph kernels.
- It outperforms many other graph networks on difficult problems