Graph Convolutional Networks

https://tkipf.github.io/graph-convolutional-networks/

GCN (Intuition)

$$\mathcal{G} = (\mathcal{V}, A)$$

A的对角元素为0

Layer-wise Propagation

$$H^{(l+1)} = f\left(H^{(l)}, A\right)$$

Aggregation from Neighbors

$$f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Self-connection and Normalization

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \qquad \tilde{A} = A + I_N \qquad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

$$\overline{\mathbf{h}}_i^{(l)} \leftarrow \frac{1}{d_i + 1}\mathbf{h}_i^{(l-1)} + \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}}\mathbf{h}_j^{(l-1)}$$

GCN

Graph Laplacian

$$\bar{L} = D - A$$

Symmetric Normalized Laplacian

$$L = D^{-\frac{1}{2}} \bar{L} D^{-\frac{1}{2}} = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$$ *式

Graph Fourier Transform

$$\hat{x} = U^T x, \quad x = U\hat{x}$$

Graph Convolution CONVOLUTION THEOREM

$$x *_{\mathcal{G}} g = U\left((U^T x) \odot (U^T g)\right) = U g_\theta U^T x$$

避免做特征值分解求U

Approximation Defferrard et al. 2016

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \qquad \tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - I_N$$

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x \qquad \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

Chebyshev Polynomial

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

$$T_0(x) = 1, \; T_1(x) = x$$

GCN

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x \qquad \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

拉普拉斯矩阵最大的特征值(的特性<=2)

$K = 1, \lambda_{\max} \approx 2$ $\Big\downarrow$ 假设1

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

上页*式

$\theta = \theta'_0 = -\theta'_1$ $\Big\downarrow$ 假设2

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x$$

$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ $\Big\downarrow$

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

Other Methods

- Graph-CNN-C / S / F:
  - C: Chebyshev 切比雪夫近似
  - S: Spline $g_\theta(\Lambda) = B\theta$
  - F: Fourier 最原始的版本

SGC

去掉了非线性函数

$$\hat{\mathbf{Y}} = \mathrm{softmax}\left(\mathbf{S}\ldots\mathbf{SSX}\boldsymbol{\Theta}^{(1)}\Theta^{(2)}\ldots\Theta^{(K)}\right)$$

$$\hat{\mathbf{Y}}_{\mathrm{SGC}} = \mathrm{softmax}\left(\mathbf{S}^{K}\mathbf{X}\Theta\right)$$

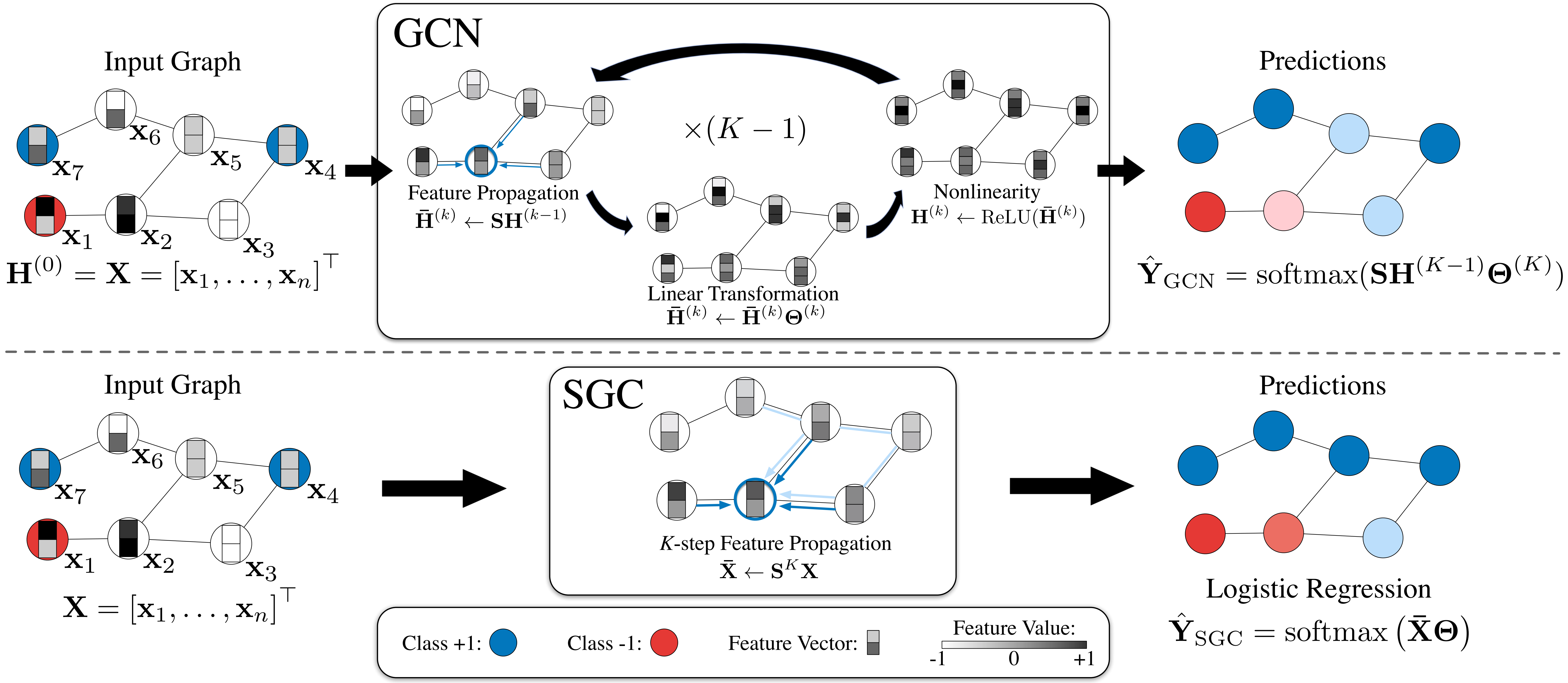

Figure 1. Schematic layout of a GCN v.s. a SGC. *Top row:* The GCN transforms the feature vectors repeatedly throughout K layers and then applies a linear classifier on the final representation. *Bottom row:* the SGC reduces the entire procedure to a simple feature propagation step followed by standard logistic regression.

Wu et al. 2019. Simplifying Graph Convolutional Networks

Laplacian Smoothing

$$\hat{\mathbf{y}}_i = (1 - \gamma)\mathbf{x}_i + \gamma \sum_j \frac{\tilde{a}_{ij}}{d_i} \mathbf{x}$$

$$\hat{Y} = X - \gamma \tilde{D}^{-1} \tilde{L} X = \left(I - \gamma \tilde{D}^{-1} \tilde{L} \right) X$$

FastGCN

FastGCN 采样部分节点替代所有节点（mini-batch），适用于超大密集图

GraphSage 采样部分节点，适用于超大稀疏图

$$H^{(l+1)}(v,:)=\sigma\left(\frac{\overline{n}}{\underline{t_l}}\sum_{j=1}^{t_l}\hat{A}\left(v,\underline{u_j^{(l)}}\right)H^{(l)}\left(u_j^{(l)},:\right)W^{(l)}\right),\quad l=0,\ldots,M-1$$

—是sample出来的

sample的数量

Thanks!