Semi-Supervised Classification with Graph Convolutional Networks

Team 35

Chirag Shameek Sahu (2019102006)

Ashish Gupta (2019101061)

Isha Gupta (2019101111)

Dinesh Garg (2019101085)

Introduction

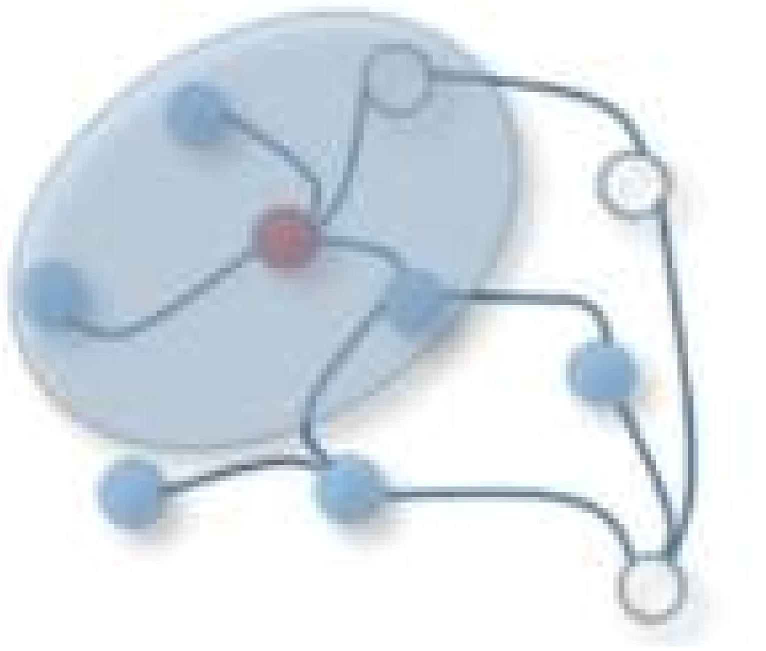
We consider the task of classifying nodes in a graph, where only a small portion of nodes are labelled.

Explicit graph-based regularization can be done e.g., by using a graph Laplacian term in the loss function.

$$L = L_0 + \lambda L_{reg}$$

$$L_{reg} = \Sigma_{ij} A_{ij} \left|\left| f(X_i) - f\left(X_j\right) \right|\right|^2 = f(X)^T \Delta f(X)$$

Here, L_0 denotes the supervised loss with respect to the labeled part of the graph, $f(.)$ can be a neural network like differentiable function, λ is the weighing factor, and X is a matrix of node features X_i . The formulation relies on the assumption that connected nodes in the graph are likely to share labels. This might restrict the modelling capacity.



Fast Approximate Convolutions on Graphs

We consider a multi-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule

$$H^{(I+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(I)}W^{(I)})$$

Here $\tilde{A} = A + I_N$ is the adjacency matrix of graph G with added self-connections. $W(I)$ is layer specific trainable weight matrix.

$H^{(I)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the I^{th} layer, $H^{(0)} = X$

Spectral Graph Convolution

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in \mathbb{R}^n$ with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \in \mathbb{R}^n$ i.e.,

$$g_\theta * x = U g_\theta U^T x$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $\mathrm{L} = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$, $U^T x$ being the graph Fourier transform of x. g_θ is a function of eigenvalues of L, i.e., $g_\theta(\Lambda)$. Evaluating the above equation is computationally expensive, as multiplication with U is $O(N^2)$. Furthermore, computing the eigen decomposition of L in the first place might be prohibitively expensive.

Spectral Graph Convolution

$g_\theta(\lambda)$ can be well approximated by a truncated expansion in terms of Chebyshev's polynomials $T_k(x)$ up to K^{th} order.

$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$.

$$g_{\theta'}(\lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\lambda})$$

with a rescaled $\tilde{\lambda} = \frac{2}{\lambda_{max}} \lambda - I_N$. $\theta' \in \mathbb{R}^k$ is the vector of Chebyshev coefficients.

$$g_{\theta'} * x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x$$

where, $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$

Layer wise Linear Model

A neural network model based on graph convolutions can therefore be built by stacking multiple convolution layers, each layer followed by a point-wise non-linearity. Let $K = 1$, a function that is linear with respect to L. We approximate $\lambda_{max} \approx 2$, then

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

In practice, it can be beneficial to constrain the number of parameters

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

$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvectors in the range $[0, 2]$. Repeated application of this operator can therefore lead to exploding/vanishing gradients. Renormalization trick:

$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}}$. We can generalize this to a signal $X \in \mathbb{R}^{N \times C}$ with C input channels and F filters or feature maps.

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

Where $\theta \in \mathbb{R}^{C \times F}$ is a matrix of filter parameters and $\mathrm{Z} \in \mathbb{R}^{N \times F}$ is convolved signal matrix.

Semi-supervised node classification

$$=x^T\left(I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)x$$

$$= \sum_{i\in V} x(i)^2 - \sum_{(i,j)\in E} \frac{2x(i)x(j)}{\sqrt{d(i)d(j)}}$$

$$= \sum_{(i,j)\in E} \left(\frac{x(i)}{\sqrt{d(i)}} - \frac{x(j)}{\sqrt{d(j)}}\right)^2 \geq 0$$

$x^T(I - B)x \geq 0$ implies

$$x^T Bx \leq x^T x \Rightarrow x^T Ix + x^T Bx \leq 2x^T x \Rightarrow \frac{x^T(I + B)x}{x^T x} \leq 2$$

Semi-supervised node classification

We consider a two-layer GCN. We first calculate $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ in a pre-processing step. The forward model takes the form:

$$Z = f(X, A) = softmax\left(\hat{A}ReLU\left(\hat{A}XW^{(0)}\right)W^{(1)}\right)$$

Here, $W^{(0)} \in \mathbb{R}^{C \times H}$ is input-to-hidden weight matrix for a hidden layer with H feature maps. $W^{(0)} \in \mathbb{R}^{H \times F}$ is a hidden-to-output to weight matrix.

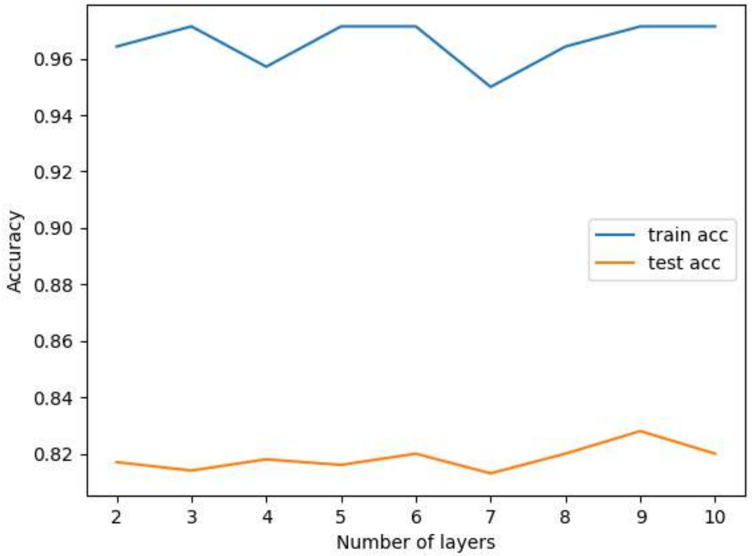
$$L = -\sum_{I \in Y_L} \sum_{f=1}^{F} Y_{If} Z_{If}$$

Work Done

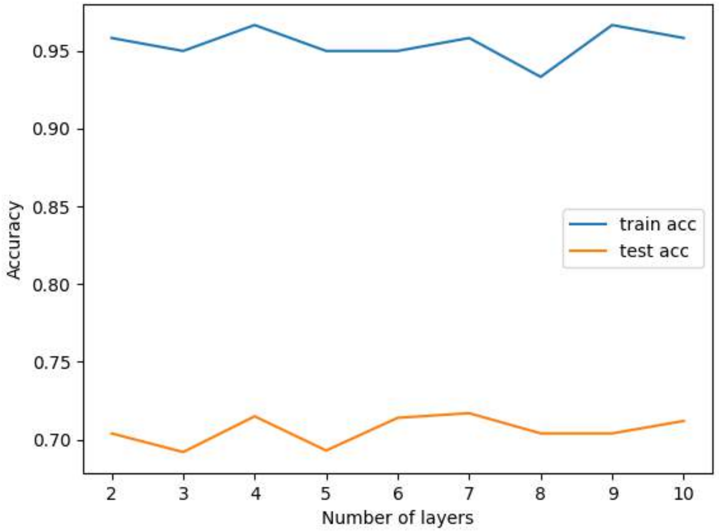
- Understanding the Paper
- Writing the code for the label propagation algorithm
- Plotting the accuracy vs change in number of layers of GCN on

1. Cora
2. Citeseer
3. Pubmed

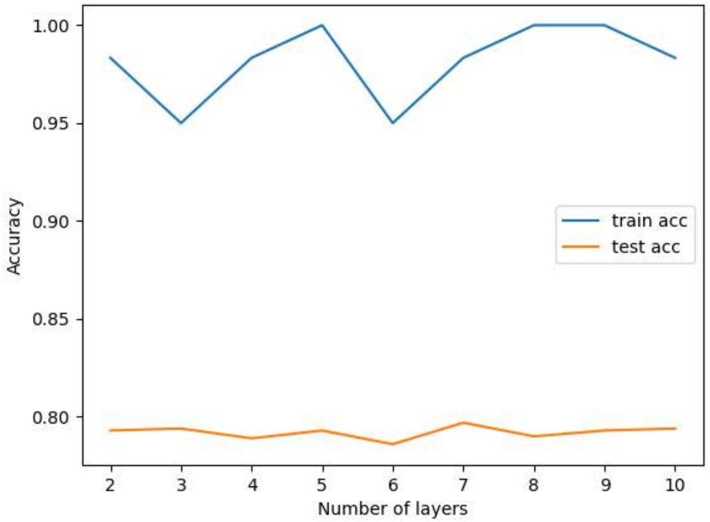
GCN & Depth



Cora



Citeseer



Pubmed

THANK YOU