Notes on:
GCN: Semi-Supervised Classification with Graph Convolutional Networks

September 5, 2024

1 Abstract

This work presents a scalable approach for semi-supervised learning on graph-structured data based on an efficient variant of convolutional neural networks which operate directly on graphs. This model scales linearly in the number of graph edges and learns hidden layer representations encoding both local graph structure and feature nodes.

2 Introduction

Consider the problem of classifying nodes in a graph. This problem can be framed as graph-based semi-supervised learning, where label information is moothed over the graph via some form of explicit graph-based regularisation e.g. using a graph Laplacian regularization term in the loss function:

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg}, \text{ with } \mathcal{L}_{reg} = \sum_{i,j} A_{i,j} \| f(X_i) - f(X_j) \|^2 = f(X)^T \Delta f(X) \quad (1)$$

Here, $\mathcal{L}_0$ denotes the supervised loss w.r.t. (with respect to) the labeled part of the graph, $f(\cdot)$ can be a neural network-like differentiable function, λ is a weighing factor and X is a matrix of node feature vectors X_i. $\Delta = D - A$ denoting the unnormalised graph Laplacian of an undirected graph $A \in \mathbb{R}^{N \times N}$ being the adjacency matrix and D the degree matrix where $D_{ii} = \sum_j A_{i,j}$.

In this work, the graph structure is encoded directly using a neural network model $f(X, A)$ and train on a supervised target $\mathcal{L}_0$ for all nodes with labels, thereby avoiding explicit graph-based regularisation in the loss function. Conditioning $f(\cdot)$ on the adjacency matrix will allow the model to distribute gradient information from the supervised loss enabling to learn representations of nodes both with and without labels.

3 Fast Approximate Convolutions on Graphs

A graph-based neural network model $f(X, A)$ is now proposed. A multi-layer Graph-Convolutional Network(GCN) with the layer-wise propagation rule:

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

Here, $\tilde{A} = A + I_N$ is the adjacency matrix with added self-connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{i,j}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function. $H^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$.

4 Spectral Graph Convolutions

A spectral convolution on graph is considered, this is defined as the multiplication of a signal $x \in \mathbb{R}^N$ (a scalar for every node) with a filter $g_\theta = \text{diag}(\theta)$ parameterised by $\theta \in \mathbb{R}^N$ in the Fourier domain, i.e.:

$$g_\theta \star x = U g_\theta U^T x \tag{3}$$

Where U is the matrix of eigenvectors of the normalised graph Laplacian $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T$, with a diagonal matrix of its eigenvalues Λ and $U^T x$ being the graph fourier transform of x.

Evaluating Eq. 3 is computationally expensive, as multiplication with the eigenvector matrix U is $\mathcal{O}(N^2)$. Furthermore, computing the eigendecomposition of L might be prohibitevely expensive for large graphs.

To circumvent this problem $g_\theta(\Lambda)$ can be well- approximated by a truncated expansion in terms of Chebsyshev polynomials $T_k(x)$ up to K^{th} order:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \tag{4}$$

with a rescaled $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$. λ_{max} denotes the largest eigenvalue of L. $\theta' \in \mathbb{R}^K$ is now a vector of Chebyshev coefficients.

$$\dots \tag{5}$$

After a few complicated mathematical tricks, it ends up with a generalised definition of the method which was BASED in a spectral method but ends up being a special case of a spatial method.

It can be generalised this definition to a signal $X \in \mathbb{R}^{N\times C}$ with C input channels and F feature maps as follows:

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta \tag{6}$$

where $\Theta \in \mathbb{R}^{C\times F}$ is now a matrix of filter parameters and $Z \in \mathbb{R}^{N\times F}$ is the convolved signal matrix. This filtering operation has complexity $\mathcal{O}(|\mathcal{E}|FC)$, as

$\tilde{A}X$ can be efficiently implemented as a product of a sparse matrix with a dense matrix.