Directed Graph Neural Networks

Stefan Dernbach
University of Massachusetts, Amherst
College of Information and Computer Sciences
dernbach@cs.umass.edu

***Abstract*— A modification of a standard graph neural network to use the directed nature of edges in many graphs improves accuracy.**

I. INTRODUCTION

Graphs are a datastructure used to represents objects and relationships. We define a simple graph $G = (V, E)$ to be a set of vertices (nodes) $V = \{v_0, v_1, ..., v_{N-1}\}$ and a set of edges $E = \{(v_i, v_j) \ : \ v_i, v_j \in V\} \subset V \times V$ representing the objects and relationships respectively. A directed graph imposes an ordering on a pair of nodes such that (v_i, v_j) denotes an edge pointing from v_i to v_j. This ordering is often useful as it further describes the relationship between the objects.

Several matrices are commonly used to represent aspects of the graph. The adjacency matrix:

$$\mathbf{A}_{ij} \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

encodes the geometry of the graph in matrix form. For an undirected graph the matrix $\mathbf{A}$ is symmetric and the degree matrix $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}[i, i] = \sum_i \mathbf{A}[i] = d(v_i)$. The combinatorial Laplacian matrix, $\mathbf{L}$, combines the degree and adjacency matrices: $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The symmetric normalized Laplacian is given as $\mathbf{L}' = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

In a directed graph, $\mathbf{A}$ is often asymmetric and we construct two degree matrices $\mathbf{D}_{in}$ and $\mathbf{D}_{out}$ where $\mathbf{D}_{in}[i, i] = \sum_j \mathbf{A}[j, i]$ and $\mathbf{D}_{out}[i, i] = \sum_j \mathbf{A}[i, j]$ count the number of incoming and outgoing edges of each vertex in the graph respectively.

II. GRAPH CONVOLUTION NETWORKS

Graph convolution layers are a fairly recent development in deep learning utilizing the graph underlying the data matrix to structure how the data is processed. The work so far on graph neural networks have focused primarily on undirected graphs. This brief focuses on graph convolution networks (GCN) [3]. Graph convolution networks were proposed as a first order approximation to localized spectral filter networks [2] which in turn were a local approximation to spectral networks [1]. Spectral networks use the graph Fourier transform to perform a graph convolution in the frequency domain of the graph a la the convolutional theorem. Localized spectral filter networks, approximate the graph spectra using Chebyshev polynomials. These polynomials have local support on the graph which transformed the global convolution of spectral networks into a local convolution; an nth order polynomial approximation has support from the n-hop neighborhood around a node.

GCNs forgo approximating the graph spectra for a purely spatial approach to a convolution based on using the graph Laplacian to diffuse information in the vertex domain. Additionally, GCNs use an augmented graph, where a self loop is added to each node. Given the augmented adjacency matrix: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, augmented degree matrix: $\tilde{\mathbf{D}} = diag(\tilde{\mathbf{A}}\mathbf{1})$, the GCN takes in features $\mathbf{X}$ and outputs $\mathbf{Y}$ according to:

$$\mathbf{Y} = \mathrm{h}(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}). \tag{2}$$

The function h is a nonlinearity such as a ReLU and the weights $\mathbf{\Theta}$ are learned by the neural network via backpropagation.

III. DIRECTED GRAPH NETWORKS

We look at two approaches to graph convolution networks designed specifically for directed graphs. The first uses a linear combination of the adjacency matrix of a directed graph and its transpose to diffuse information across the graph:

$$\mathbf{Y} = \mathrm{h}((\lambda\mathbf{A} + (1 - \lambda)\mathbf{A}^T)\mathbf{X}\mathbf{\Theta}) \tag{3}$$

The parameter λ balances the importance of passing information forward along edges of a graph and backwards along edges. We also optionally look at an augmented version of the graph with self loops on nodes which is necessary in this formulation for a node to retain information about its own input features.

Our second approach is a direct adaptation of GCNs. We replace the augmented diffusion matrix in 2: $\mathbf{I} + \tilde{\mathbf{L}}' = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with the equivalent directed version given by:

$$\mathbf{I} + \tilde{\mathbf{L}}_{dir} = \tilde{\mathbf{D}}_{in}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}_{out}^{-1/2}. \tag{4}$$

We then modify the structure of the neural network to process the input in two directed graph layers in parallel as show in figure 1. The two layers compute the following:

$$\mathbf{Y}_F = \quad \mathrm{h}((\tilde{\mathbf{D}}_{in}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}_{out}^{-1/2})\mathbf{X}\mathbf{\Theta}) \tag{5}$$

$$\mathbf{Y}_B = \quad \mathrm{h}((\tilde{\mathbf{D}}_{in}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}_{out}^{-1/2})^T\mathbf{X}\mathbf{\Theta}). \tag{6}$$

The layer in (5) diffuses features in the graph following the direction of edges, while (6) diffuses features against the direction of the edges. The output of the two layers is

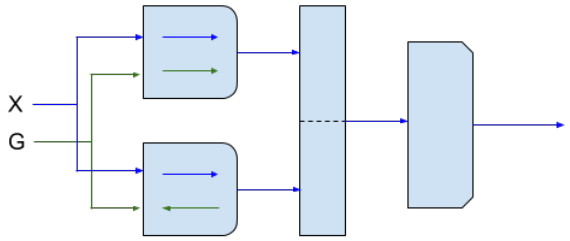

Fig. 1. The directed graph convolution network architecture.

TABLE I
EXPERIMENTAL RESULTS

Network	Cora	CiteSeer
GCNN	83.41 ± 2.69	73.35 ± 2.18
Parallel GCN	86.87 ± 0.77	
Linear Adjacency Combination	80.46 ± 1.09	70.56 ± 0.98
Augmented Linear Adjacency	85.79 ± 2.17	
Directed GCN	87.89 ± 2.43	

concatenated and fed forward into subsequent layers in the network.

IV. EXPERIMENTAL EVALUATION

We compare the undirected version of the GCN with our pair of directed methods on the Cora citation dataset [4]. To account for the increased capacity of having parallel layers in the directed GCN, we also compare against a network with parallel undirected GCN layers.

The Cora dataset consists of 2708 papers linked by 5429 edges denoting a citation from one paper to another. Each paper has a binary vector describing it which represents the presence or absence of 1433 unique words. The goal of the experiment is to classify each paper into one of 7 categories. The neural networks are given the labels for 80% for training, 10% are reserved for model validation, and models are evaluated based on their accuracy predicting the remaining 10%. Each model is trained for 400 iterations.

REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[3] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[4] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.