Graph Kernel Layers

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

***Abstract*— A comparison of different graph kernels used for diffusing features in a graph neural network.**

I. INTRODUCTION

Graphs are a datastructure used to represents objects and relationships. Mathematically, 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. 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. Each vertex in the graph corresponds to a row and column in the adjacency matrix and edges in the graph are represented by a 1 in the corresponding row-column index. The degree matrix $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}_{ii} = \sum_i \mathbf{A}_i = d(v_i)$. 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 $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Graph convolution layers are a fairly recent development in deep learning. They allow data on graphs to be processed in a way that utilizes the graph structure. Graph convolutional layers come in two forms. The first, spectral graph layers [2], use the graph Fourier transform defined by the eigenvectors of the graph Laplacian to process data in the graph frequency domain. We don't deal further with these methods in this paper.

The second type of layer, spatial graph convolution layers uses the adjacency and Laplacian matrices to diffuse information across the graph as its processed. Many spatial graph convolutional neural network layers can be generalized to the form $\mathbf{Y} = \mathcal{K}(G)\mathbf{X}\mathbf{\Theta}$ where the output of the layer, $\mathbf{Y} \in \mathcal{R}^{N \times k_{out}}$, is a product of a kernel on the nodes of the graph, the input signal $\mathbf{X} \in \mathcal{R}^{N \times k_{in}}$, and the learnable weights $\mathbf{\Theta}$. The shape of the weights $\mathbf{\Theta}$ also vary to match the shape of $f(\mathbf{A})\mathbf{X}$ and transform it into the desired output shape.

The kernel matrix $\mathcal{K}(G)$ determines the type of graph layer. Graph Convolution Neural Networks [4] use $\mathcal{K}(G) = (\mathbf{D}{+}\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}{+}\mathbf{I})(\mathbf{D}{+}\mathbf{I})^{-\frac{1}{2}}$. This can be equivalently viewed as defining a new graph $\tilde{G}$ identical to the old graph but with a self-loop on every node and then using the kernel $\mathbf{I} + \mathbf{L}_{\tilde{G}}$.

Diffusion Convolution Neural Networks [1] vary the pattern slightly by using a stacked tensor of kernels such that $\mathcal{K}(G) = [(\mathbf{A}\mathbf{D}^{-1})^0, (\mathbf{A}\mathbf{D}^{-1})^1, ..., (\mathbf{A}\mathbf{D}^{-1})^k]$. Additionally, the original paper for this layer uses an element wise product between the features and the layer weights rather than a dot product.

II. Node Kernels

There exists a wide breadth of spatial graph kernels that can be drawn on to develop new graph neural network layers. This section defines a selection of these kernels [3].

A. *Exponential Diffusion Kernels*

The standard exponential diffusion kernel [5] is defined as:

$$\mathcal{K}_{ED}(G) = \sum_{k=0}^{\infty} \frac{\alpha^k \mathbf{A}^k}{k!} = \exp(\alpha\mathbf{A}). \tag{2}$$

B. *Laplacian Exponential Diffusion Kernel*

As will be seen to be common in these kernels, the Laplacian is often substituted for the adjacency matrix. Beginning with the exponential diffusion kernel and replacing $\mathbf{A}$ with $-\mathbf{L}$ produces the Laplcian exponential diffusion [5], [8] kernel:

$$\mathcal{K}_{LED}(G) = \sum_{k=0}^{\infty} \frac{-\alpha^k \mathbf{L}^k}{k!} = \exp(-\alpha\mathbf{L}). \tag{3}$$

C. *Von Neumann Diffusion Kernel*

Removing the discounting scheme from the exponential diffusion kernel leads to the von Neumann kernel [7]:

$$\mathcal{K}_{VN}(G) = \sum_{k=0}^{\infty} (\alpha\mathbf{A})^k = (\mathbf{I} - \alpha\mathbf{A})^{-1} \tag{4}$$

D. *Regularized Laplacian Kernel*

Again the replacement of $\mathbf{A}$ with $-\mathbf{L}$ gives rise to a new kernel. The regularized Laplacian kernel [8]is defined as:

$$\mathcal{K}_{RL}(G) = \sum_{k=0}^{\infty} (-\alpha\mathbf{L})^k = (\mathbf{I} + \alpha\mathbf{L})^{-1} \tag{5}$$

E. *Commute Time Kernel*

The commute time kernel [6] is given by:

$$\mathcal{K}_{CT}(G) = \mathbf{L}^{+}. \tag{6}$$

$\mathbf{L}^{+}$ denotes the Moore-Penrose pseudo-inverse of the Laplacian matrix.

TABLE I
EXPERIMENTAL RESULTS

Kernel	Cora	CiteSeer
GCN	83.41 ± 2.69	73.43 ± 2.42
DCNN	84.12 ± 1.86	78.00 ± 2.48
Exponential Diffusion ($\alpha = 10^{-6}$)	71.06 ± 0.91	70.67 ± 2.70
Laplacian Exponential ($\alpha = 10^{-2}$)	70.74 ± 1.48	72.57 ± 1.32
Laplacian Exponential	15.07 ± 3.15	69.67 ± 1.87
Von Neumann	78.49 ± 3.64	71.03 ± 4.0
Regularized Laplacian	77.66 ± 1.59	69.57 ± 1.61
Commute Time	81.00 ± 2.85	70.73 ± 2.29
Augmented Commute Time	88.51 ± 01.15	79.01 ± 0.67
Truncated Commute Time (2)	87.39 ± 2.06	76.75 ± 3.26
Truncated Commute Time (3)	$\mathbf{89.86 \pm 1.23}$	$\mathbf{79.42 \pm 2.11}$
Truncated Commute Time (4)	88.29 ± 2.47	76.41 ± 1.60

III. EXPERIMENTAL EVALUATION

We compare graph layers using the various given kernels as well as GCN and DCNNs on the Cora citation dataset. The 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.

Each kernel forms the diffusion matrix for a single layer neural network. If the kernel has an α parameter, it is included as an additional learnable weight in the network. Kernels that use the graph Laplacian are evaluated using both the combinatorial Laplacian, and the symmetric normalized Laplacian. Additionally, an augmented form of the Laplacian with self loops on the nodes (as in GCNs [4]) is tested. The results are given in Table I.

The results for DCNN are given for 2 hops, which could be reformulated as a multilayer DCNN. This effectively gives it an advantage over the other models which are single layer networks. It is unknown at the moment why the exponential kernels only match the baseline of randomly selecting a class label. This will require further work to determine what is happening within the neural network. One planned approach is to select a constant α parameter to determine if that improves the model. The commute time kernel does exceedingly well, especially the augmented form. The normalized form however, does not work out at all, normalizing it likely has an effect on the inverse. Overall the results give a good reason to look further into the commute time kernel.

REFERENCES

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[2] 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.

[3] Francois Fouss, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of five graph kernels on a collaborative recommendation task. In *IEEE International Conference on Data Mining (ICDM)*, 2006.

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

[5] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pages 315–322, 2002.

[6] Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *European Conference on Machine Learning*, pages 371–383. Springer, 2004.

[7] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[8] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003.