Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Lokesh Kumar T
Department of Electrical Engineering
Indian Institute of Technology Madras
Chennai, India
lokesh.karpagam@gmail.com

January 28, 2019

Abstract

This report we extensively deal with the concepts associated with **Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering**. We summarize the paper and comment on the novelty and innovation involved. Moreover, we go ahead with a detailed critique and end with providing future possible improvements and areas of research.

1 Summary of the Paper

We gloss over the mathematical and note worthy details presented in the paper. This section is ordered in this fashion

1.1 Introduction

Graphs are data structures that can effectively model high dimension dependencies with nodes of the graph representing the object and the edges representing the relationship between the nodes. Convolutional Neural Networks are efficient architectures that extract meaning full information in large scale and high dimensional datasets. The ability of CNNs has lead breakthorughs in computer vision, speech processing etc.

1.2 Prerequisites

Undirected Graph Adjacency Matrix:

$$A_{ij} = a_{ij}\ if\ (i,j) \in \mathbb{E}, \quad else\ 0 \tag{1}$$

Signal from the graph: Vector representing value for each node in G.

$$x \in \mathbb{R}^n = (x_1, x_2, ..., x_n)^T \tag{2}$$

Degree Matrix (D): Diagonal Matrix

$$D_{ii} = d_{ii} \;\; \forall \; i \in \mathbb{V} \;\; (degree\ of\ node\ i) \tag{3}$$

Laplacian Matrix (L):

$$L = D - A \tag{4}$$

Normalized Laplacian Matrix Definition

$$L = I_n - D^{-\frac{1}{2}} W D^{\frac{1}{2}} \tag{5}$$

1.3 Proposed Technique

The paper splits the spectral based graph convolutional networks into three fundamental steps (i) local convolutional filters, (ii) a graph coarsening and (iii) graph pooling operation.

1.3.1 Learning Fast Localized Spectral Filters

We must remember that convolutional theorem states that convolution are linear operators that diagonalize the Fourier basis (here which are represented as eigenvectors of Laplacian). We know that time domain convolution of two time domain signals can be written as a product operation in fourier domain. We draw support from this analogy to draw our conclusion in the case of graph convolution.

As L is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors and real non-negative eigenvalues. Eigenvalues are called frequencies of the graph and eigenvector set is called Fourier modes of the graph.

The eigenvalue decomposition of Laplacian of the graph

$$L = U \Lambda U^T \tag{6}$$

Graph Fourier Transform definition: Let x be a signal (n-dim) then its graph Fourier transform is defined as,

$$\hat{x} = U^T x \in \mathbb{R}^n \tag{7}$$

Inverse Fourier transform definition:[1]

$$x = U \hat{x} \in \mathbb{R}^n \tag{8}$$

Graph Convolutional Operator in Fourier Domain: (drawing analogy form time domain signal convolution)

$$x \star_G y = U((U^T x) \odot (U^T y)) \tag{9}$$

[1]Note that $U^T U = I$ as U is the eigenvector matrix of a symmetric PSD matrix (L).

Non paramteric filter g_θ filtering operation can be defined as,

$$y = g_\theta(L)x = U g_\theta(\Lambda) U^T x \tag{10}$$

As localization and lesser learning complexity are very important in CNNs we must constraint the parameters by following a parameterization strategy. One such parameterization technique is Polynomial Parameterization of filters.

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^K \tag{11}$$

Using this we can include spatial sharing and the learning complexity is still $\mathcal{O}(K)$ (similar to classical CNNs). This procedure still involves matrix multiplication with U (Fourier Basis) which is $\mathcal{O}(n^2)$.

$$g_\theta(L) = \sum_{k=0}^{K-1} \theta_k T_k(L) \tag{12}$$

In the above paramterization $T_k(x)$ denotes the Chebyshev polynomial of kth order. By noting $(U\Lambda U^T)^k = U\Lambda^k U^T$, we can verify that the expression is K-localized since its a Kth- order polynomial of the Laplacian and depends only on nodes that are maximum K steps away from the central node.

Filters can be learnt using the backpropogation algorithm which involves K sparse matrix-vector multiplication and one dense matrix-vector multiplication.

1.4 Graph Coarsening

Graph pooling requires similar vertices clustered together. Repeating the same for multiple layers is equivalent to multi-scale clustering of the graph and it must respect geometric structures. The graph clustering algorihtm that is well known is the *spectral clustering.* In this paper, the authors have made use of coarsening phase of the *Graclus multilevel clustering algorithm* which has shown to be more effective on a variety of graphs.

1.5 Fast Pooling of Graph Signals

Coarsening leaves the vertices arranged in a non-meaningful way. Therefore application of pooling operation would lead to unreasonable demand of memory and inefficiency (not parallelizable). We need to arrange the vertices in a fashion so that graph pooling becomes efficient 1D pooling. The novelty is present in the re-indexing proposed which evades the creation of tables and makes the algorithm more efficient in terms of memory and implementation. Follow these steps (i) create a balanced binary tree and (ii) rearrange the vertices. The structure of the balanced binary tree (i) regular nodes or (ii) one singleton and a fake node as children. Pooling such a rearranges signal becomes analog to pooling 1D signal and hence can benefit from massive parallel architectures like GPUs.

1.6 Numerical Experiments

This paper compares classical CNNs and GCNs in case of MNIST dataset. An 8-NN graph of 2D grid which produces n=976 nodes and 3198 edges. Deep GCNs outperformed shallow GCNs. Classical CNNs are faster in both CPU and GPU than GCNs, but the fraction of speed up obtained in case of GCNs is greater. When the dataset is changed to 20NEWS, GC32 outperforms all the models except multinomial naive bayes. Its also observed that as the number of words increases the parameterized Chebyshev takes less time which compared to non-parameterized filters (computational complexity).

2 Critique

2.1 Criticism and List of shortcomings

2.1.1 Inability to perform Transfer Learning

Transfer learning plays a very important role in deep learning today as it enables to achieve high accuracy in small datasets. In this paper, the graph neural network uses Laplacian of the graph in Chebyshev parameterization which is unique to a graph. This means we cannot perform transfer learning from one dataset to another as the graph Laplacian changes. The inclusion of strategies which enables this transfer learning should be explored in future. The algorithm must try to imbibe some general and global features shared by all graphs so that extension from one graph to another is possible.

2.1.2 Experiments Chosen

The experiments chosen in the paper are MNIST dataset classification and 20NEWS dataset. The experiments could have been done in some established graph datasets so that the results could be better understood and well scrutinized. The margins of defeat are quite small which seems to suggest contrary to what the authors are claiming. Moreover this seems to suggest that the contribution is much more in the side of computational efficiency than the effectiveness. This can also be inferred from the GPU and CPU speeds given in the results. The speedup obtained in the case of graph CNN was 8.00x when compared to speedup of 6.77x in classical CNN. Experiments related to scalability should also be tested to determine its practical use.

2.1.3 Theory presentation and Readability

The presentation of theory was rather shallow which meant a lot of outside reading is required in understanding the concepts. The notational consistency could have been respected in the Numerical Experiments section where a lot of terms were introduced. Background information and more of illustrative depiction would have helped the reader to have a smooth understanding of the subject.

2.2 Ways to extend and Future possible research areas

2.2.1 Extension of support to Dynamic Graphs

The formulation breaksdown when the graph under consideration is not static graph rather dynamic graph. Dynamic graphs are graphs with dynamic structures (changing structures). The aim of such a graph is to capture and understand dynamical and changing systems (multi-stage, time evolving networks). Dealing with static networks give advantages of stability and modelling feasibility, but dealing with dynamic graphs are more involved. The edges and nodes of a dynamic graph appear and disappear which means the graph adjacency matrix and graph laplacian changes as the graph evolves. This also has a wide variety of applications like in digital communication graphs, brain modelling graphs, molecular structure analysis graphs (all are dynamic graphs) etc.

2.2.2 Scalability

The current algorithms for Graph CNNs are inefficient in terms of memory and compute demand when the scale of the graph increases dramatically as in the real life case. The real life graphs extremely large in the sense that the number of nodes and the interacting edges very high. In case Facebook users graph where each user can be considered as a node, as of 2013 the number of active users are around 1100 million which makes these algorithms computationally untraceable. The scaling of Graph Convolutional networks are difficult because of its dependence on graph Laplacian matrices which can become extremely large in these cases. Moreover, batch processing cannot be applied as the graphs are not Euclidean and as metioned before Laplacian calculation is also infeasible. These problems need to be solved for putting these algorithms into practical use and hence should be considered as a future possible work.

2.2.3 Limitations on Deep Graph Convolutional Networks (GCN)

In the case of conventional deep neural networks several hundred of layers can be stacked and trained to perform a task (ResNets in computer vision). The deeper networks tend to perform better[2] because of the fact that the increased parameters in them help them to possess greater representation ability. As noted in this paper and several other graph neural networks papers the networks are always shallow. The work in [1] show that graph convolution in GCNs are special form of *Laplacian Smoothing* which mixes features of a vertex and its nearby neighbors. The smoothing operations make the clusters similar, thus making the tasks for GCN easier. When over-smoothing occurs the clusters become indistinguishable and hence performance considerably drops. Future work along this direction will contribute to understanding of GCNs in general.

[2]Overfitting and vanishing gradients are problems faced when training deep networks.

2.2.4 Need for Labelled data

In the work [1] it has been established understanding of GCNs needs development and GCNs are label data intensive. The working mechanisms of the GCN model for semisupervised learning are not clear and as the training requires a lot of labelled data, this defeats the purpose. As we have seen previously that deep GCNs cannot perform well, even very shallow GCNs suffer from localized nature of convolutional filter. When only few labels are given, a shallow GCN cannot propogate the labels to the entire data graph. The extensive need for validation data for model selection and hyperparameter tuning also should be minimized.

2.2.5 Receptive Field of Shallow GCNs

As we have seen from Laplacian over smoothing problem we cant have deep GCN networks, but even shallow GCNs suffer from some limitations from the convolution operations. The localized nature of convolution means the receptive field of shallow GCNs are low and hence can cause performance downgrades. Therefore either this problem or the realization of deep GCNs must be enabled to effective utilization of GCNs in practical applications.

References

[1] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[2] Defferrard, M., Bresson, X., and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, 3844–3852. Curran Associates, Inc.

[3] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, [online] Available: https://arXiv:1609.02907.

[4] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. arXiv preprint arXiv:1806.01973 (2018)

[5] Jie Zhou , Ganqu Cui , Zhengyan Zhang , Cheng Yang, Zhiyuan Liu, Maosong Sun. 2019. Graph Neural Networks: A Review of Methods and Applications

[6] Ananth Kalyanaraman. 2017. Advances in Algorithms and Applications for Dynamic Graphs slides

[7] Xavier Bresson. 2018. Convolutional Neural Networks on Graphs. YouTube