Project Draft for Semi-Supervised Classification with Graph Convolutional Networks

Zhisheng Xu and Goutham Pakalapati
{zx41, gpakal2}@illinois.edu

Group ID: 28
Paper ID: 11
Presentation link: N/A
Code link: N/A

1 Introduction

The Semi-Supervised Classification with Graph Convolutional Networks (Kipf and Welling, 2017) aims to solve the problem of semi-supervised classification on graph-structured data. GCNs are engineered to take advantage of the data's local structure within graphs by utilizing convolutional layers on the adjacency matrix of the graph. It develops a scalable and efficient method that can accurately classify nodes or predict labels for unseen data points in the graph which has limited amount of labeled data.

2 Scope of reproducibility

The proposed GCN model can achieve state-of-the-art performance on semi-supervised node classification tasks by effectively leveraging both labeled and unlabeled data through graph-based regularization. Specifically, the proposed GCN model outperforms previously proposed methods on three benchmark datasets (Cora, Citeseer, and Pubmed) in terms of classification accuracy while maintaining the competitive runtime speed.

3 Methodology

To reproduce the results, we will utilize both the datasets and code from the original author. The code has undergone minor modifications to make it compatible with the latest TensorFlow and Scipy packages. Additionally, we plan to further tweak the code to test different activation functions later in the project. The code was obtained from the author's GitHub repository, located at https://github.com/tkipf/gcn. By following the instructions provided in the README file, we were able to successfully execute the models using the datasets provided. The computations were performed on an RTX 3080 Ti GPU.

3.1 Model descriptions

The GCN model is a two-layer neural network of graph convolutional layers. The first layer is a graph convolutional layer with 16 hidden units followed by a ReLU (Rectified Linear Unit) activation function. The second layer is another graph convolutional layer, which serves as the output layer and has as many output units as the number of classes in the classification problem.

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf} \tag{1}$$

3.2 Data descriptions

We will be testing with 3 citation network datasets – Citeseer, Cora and Pubmed – nodes are documents and edges are citation links. These datasets are obtained from the author's GitHub repository. Following table contains statistics of the datasets.

Dataset	Nodes	Edges	Features	Classes
Citeseer	3,327	4,732	3,703	6
Cora	2,708	5,429	1,433	7
Pubmed	19,717	44,338	500	3

Table 1: Dataset statistics

3.3 Hyperparameters

The model in the paper used a learning rate of 0.01, a weight decay (L2 regularization) of 5e-4, and a dropout rate of 0.5. The model was trained for 200 epochs using the Adam optimization algorithm. Early stopping was applied to prevent overfitting, with training stopping after 10 epochs with no improvement in validation loss. These hyperparameters were determined through a combination of grid search and manual tuning, and have been shown to perform well on the benchmark datasets used in the paper.

3.4 Implementation

As mentioned previously, we plan to use the code provided by the authors to replicate the main experiments presented in the paper. Our code with the modifications for ablations will be available at `https://github.com/xuzhisheng93/cs598-dl-healthcare-proj`.

3.5 Computational requirements

The author used a 16-core Intel Xeon CPU E5-2640 v3 @ 2.60GHz with a NVIDIA GeForce GTX TITAN X. We will be running the model with AMD Ryzen 7 3700X 8-Core Processor and NVIDIA GeForce RTX 3080 Ti. In the Results section of the original paper, it states the running time on each of the datasets:

- Citeseer: 7 seconds
- Cora: 4 seconds
- Pubmed: 38 seconds

Since the power of our GPU is greater than theirs, we expect the running time to be faster than the original paper.

4 Results

4.1 Results of Reproduction

Overall, we managed to execute the original model with the provided datasets. And we are able to achieve similar results to the original paper. Our runtime speed is faster than the original paper, which is due to the fact that we are using a more powerful GPU.

Table 2: Classification Accuracy and Runtime Comparison

Dataset	Accuracy		Runtime (s)	
	Paper	Ours	Paper	Ours
Citeseer	70.3%	70.6%	7	5.3
Cora	81.5%	81.7%	4	3.4
Pubmed	79.0%	79.4%	38	12.5

We could use something like grid search on other datasets as well to further optimize some hyperparameters used in the model. Processing the graph before feeding into a GCN (e.g. adding new edges based on similarity measure)

4.2 Analysis

References

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representations (ICLR)*.