Link prediction and Node Classification in Cora citation network

Kavya Kavuri
Rutgers University - New Brunswick
kavya.kavuri@rutgers.edu

Parvathi Mahesh Hedathri
Rutgers University - New Brunswick
parvathi.mh@rutgers.edu

Sumeet Bhatnagar
Rutgers University - New Brunswick
sumeet.bhatnagar@rutgers.edu

ABSTRACT

Network Graphs have been evolving in every field and naturally, there is a strong growth in predictive tasks on graph. Graph embeddings has proven to be the very effective for the prediction tasks ranging from content recommendation to computer aided drug design. In this project we explore a variety of methods to perform node classification and link prediction using the CORA graph dataset. We compare the performance of GraphSAGE and GCN models on this dataset and survey the mathematical foundations of these networks.

KEYWORDS

GrphSAGE, GCN, embedding, node classification, link prediction

1 INTRODUCTION

Graphs are information-rich, making predictions of it requires a method to effectively grasp the information into a mathematically usable fashion. Node embeddings are very useful in prediction tasks on graphs. These node embeddings capture the high-dimensional information of nodes like their neighborhood nodes, degree etc to one-dimensional vector representation. This makes a variety of prediction task like node classification, link prediction much easier.

Complex networks can be represented as graph instances. The complex interactions between entities can be represented as edges of the graph. Complex networks in the real world can be online social networks, knowledge graphs, biological networks such as genetic interactions or protein-protein interactions, and citation networks. All of these real-world networks are dynamic in nature, nodes and edges are added or removed to the graph network. Studying the trend with which these graphs evolve has many advantages. To be more specific, studying the future associations between entities and classification of these entities holds a lot of importance. This is because the problem of link prediction and node classification in complex graphs is a challenge which is very much prevalent since the evolution of technology. Given a graph which is nothing but the abstraction of entities and their relationships of a given network, link prediction is the problem of predicting the future connections among the graph entities. Node classification is the problem of classifying the entities into classes for better understanding.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

The dataset that we are using is CORA citation dataset, which consists 2708 nodes, representing scientific publications classified into one of seven classes and 5429 edges of graph representing the links. We implement several models which use GraphSAGE node embeddings and GCN node embeddings and predict the class of the node and existence of a link given two nodes. We evaluate each model to understand which algorithms are best and appropriate to get best results for our link prediction and node classification problems.

2 RELATED WORK

Several different approaches have been utilised for applying semi-supervised learning on graphs, however they can be broadly classified into two categories - using Laplacian regularization or using graph embedding-based approaches [5].

One approach is to represent labeled and unlabeled data as vertices in a graph, with the weights of the vertices between them signifying the level of correlation between them. We can then utilise Gaussian random field to formalise the problem [7].

Another approach is to use manifold regularization to incorporate label and unlabeled data to develop a general use trainer. The theoretical basis for this method is rooted in Reproducing Kernel Hilbert spaces. Using them, the model could then handle unlabeled data effectively [1]

One notable approach utilises deep semi-supervised embeddings applied to deep multi-layer neural network architectures either on every layer or simply at the output layer of the neural network. This approach again yields competitive results compared to other semi-supervised techniques discussed above. [6]

However, newer models tend to focus more on models built on top of the skip-gram model [5]. The skip-gram model was notable in not utilizing matrix multiplication while generating its embeddings, which allowed it to train upto 100 billion words in a day. [2]

Lastly, algorithms like DeepWalk [3] and LINE [4] sample random walks made on a graph to generate embeddings for each node of the graph.

3 EXPLORATORY DATA ANALYSIS

We analyzed the dataset to find its main features. We found that the dataset is not a connected graph. The top five categories with highest degrees is shown in figure 1. There are 3563 cliques in total in the dataset. The size of the maximum clique of this dataset is 5. There are 9 cliques of size 5. All of them are in either Reinforcement Learning or Neural Networks. As seen in fig 2, the class "Neural Networks" has the most number of scientific publications followed by "Probabilistic methods" which has the next highest number of scientific papers.

The visualization of CORA citation dataset can be see in below Figures 3,4.

Categories	Paper_id	Degree
Genetic_Algorithms	35	168
Reinforcement_Learning	6213	78
Neural_Networks	1365	74
Neural_Networks	3229	65
Neural_Networks	910	44

Figure 1: Scientific paper with highest degrees

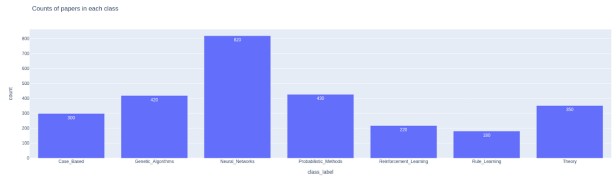

Figure 2: Counts of paper in each class

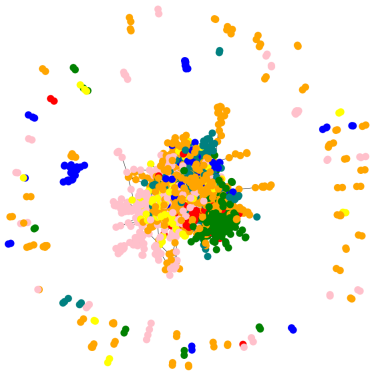

Figure 3: Plot of Scientific papers as nodes and citations as links

Figure 4: 3D plot of the graph

4 PROBLEM FORMALIZATION

4.1 Link Prediction

Each node in the graph is converted to an embedding. This is a low-dimensional vector. Any node N is represented as a vector of size 1x*dim*. Here dim is a parameter that we set.

For GraphSAGE link prediction model, we concatenate the node vectors using *multiply* operator and get a score for the existence of link between the nodes.

4.2 Node Classification

Similar to link prediction, for solving this problem as well, each node of the graph is converted to low-dimensional vector. However, the label to which the node belongs to is preserved. The embeddings are passed as inputs and the labels (in one-hot vector form) are passed as the outputs to a single layer of a conventional neural network. As you would expect in a typical multi-class classification problem, the final output layer consists of a softmax layer. For classifying a previously unknown node, we use the graphSAGE model to generate an embedding for the node in the context of the graph and then feed it to the predictor of the dense model we designed.

5 THE PROPOSED MODEL

The proposed models are:

5.1 GCN

Graph convolutional network (GCN) is a neural network that operates on graphs. It is an efficient variant of convolutional neural networks. It takes input as feature matrix for each node, adjacency matrix and performs a simple forward propagation to achieve vector representations of each node.

For node prediction, these node embeddings are passed to a keras dense layer with softmax activation to obtain the class prediction.

For link prediction, the node embeddings are concatenated using simple sum to obtain a link embedding, which then is passed to a dense layer to obtain link predictions. That is the probability of a link existing between those nodes. The aforementioned method can be mathematically written in the following way, here the $\bigoplus$ operation is the sum of two vectors:

$\hat{y} = v \bigoplus v$

5.2 graphSAGE

graphSAGE is an inductive semi-supervised learning framework. SAGE is an acronym for sampling and aggregation. The framework improves over GCN by inculcating node representation of not only the neighbors of a node, but also the neighbors of its neighbors. To reduce the overhead, it only randomly samples a fixed number of neighbors and neighbors of neighbors at every iteration for a fixed number of iterations.

The model we have designed consists of a single layer of a GraphSageModel of size 32x32 whose inputs are the sampled inputs aggregated by the generator. The model then creates the vectorized inputs and vectorized outputs which can then be fed to a single

dense layer of a neural network, whose outputs are in turn fed to a softmax output layer to allow for multi-class classification.

For link prediction, we take the node embedding vector v and perform the following operation to get the link score:

$score(\hat{y}) = v^T.v$

6 EXPERIMENTS

We have evaluated our models against four popular metrics - Accuracy, Precision, Recall and f-score. The results for different models are tabulated below.

6.1 Link Prediction

Overall, to obtain benchmarks, we performed link prediction, using three different models in total - Logistic Regression, Graph Convolutional Network and graphSAGE. The Logistic Regression model served as the baseline owing to its simplicity.

6.1.1 Logistic Regression. The simple logistic regression model was fed embeddings generated using the Node2Vec algorithm. The results obtained are as follows:

Metric	Score
accuracy	0.594
f_score	0.688
precision	0.742
recall	0.642

Figure 5: Metric results for logistic regression link prediction

6.1.2 GCN. The Graph Convolutional Network uses forward propagation to generate vector representations of the graph. We generated vectors of length 16 for each node and concatenated them to get link embeddings. These embeddings were fed to a dense layer to make the prediction,i.e., the probability of existence of a link between these nodes.

The results obtained for this model are as follows:

Metric	Score
accuracy	0.7416
f_score	0.7148
precision	0.7977
recall	0.6476
AUC	0.8406

Figure 6: Metric results for GCN link prediction

6.1.3 GraphSage. A single hidden layer of Graph convolutional network is used with kernel size 16 x 16. The output is h, which is a node representation. For the prediction, we did a dot product between source node vector and destination node vector. These are the scores for the existence of link between the source node and destination node.

We experimented with various kernel sizes and hop distances. The following results are the metric values for the optimum size 16 x 16 with hop distance 2. However this is not an exhaustive testing and the results obtained are a local optimum.

Metric	Score
accuracy	0.763
f_score	0.757
precision	0.777
recall	0.738
AUC	0.862

Figure 7: Metric results for GraphSAGE link prediction

From the combined metric results we noticed that GraphSAGE algorithm give better results as compared to GCN and Logistic regression.

Model	Accuracy	F-Score	Precision	Recall	AUC
GraphSage	0.763	0.757	0.777	0.738	0.862
GCN	0.741	0.714	0.797	0.647	0.84
Logistic Regression	0.594	0.688	0.742	0.642	0.608

Figure 8: Combined metric results for link prediction

6.2 Node Classification

Overall, to obtain benchmarks, we attempted the node classification, using three different models in total - Logistic Regression, Graph Convolutional Network and graphSAGE. The Logistic Regression model served as the baseline owing to its simplicity.

6.2.1 Logistic Regression. The simple logistic regression model was fed embeddings generated using the Node2Vec algorithm. The results obtained are as follows:

Metric	Score
accuracy	0.738
f_score	0.733
precision	0.739
recall	0.738

Figure 9: Metric results for logistic regression node classification

6.2.2 GCN. The Graph Convolutional Network uses forward propagation to generate vector representations of the graph. These representations are then fed as input to a single hidden dense layer followed by a softmax output layer for multiclass classification.

The results obtained for the algorithm are as follows:

6.2.3 GraphSage. A single hidden layer of Graph convolutional network was used with kernel size 32 x 32. To classify the results a softmax layer was used.

The number of samples count was kept at 10 for 1st hop and 5 for 2nd hop respectively. We experimented with various different values for this count, expecting an improvement or deterioration. However, this was not observed, and the results were all very similar to results given below. We believe this is because of the relatively small size of the dataset and varying the count will have a greator impact when dealing with much larger datasets.

Metric	Score
accuracy	0.7848
f_score	0.7852
precision	0.8091
recall	0.7626

Figure 10: Metric results for GCN node classification

Metric	Score
accuracy	0.806
f_score	0.81
precision	0.824
recall	0.797

Figure 11: Metric results for GraphSAGE node classification

From the combined metric results we noticed that GraphSAGE algorithm give better results as compared to GCN and Logistic regression.

Model	Accuracy	F-Score	Precision	Recall
GraphSage	0.806	0.81	0.824	0.797
GCN	0.784	0.785	0.809	0.762
Logistic Regression	0.738	0.733	0.739	0.738

Figure 12: Combined metric results for node classification

7 CONCLUSIONS AND FUTURE WORK

In this project, we used the CORA citation dataset, which consists of 2708 nodes, representing scientific publications classified into one of seven classes and 5429 edges of graph representing the links. We implement several models which use GraphSAGE node embeddings and GCN node embeddings and predict the class of the node and existence of a link given two nodes. We evaluated each model to understand which algorithms are best and appropriate to get best results for our link prediction and node classification problems. To understand which model was best suited for our prediction and classification problem we use metrics like accuracy, f_score, precision, recall and AUC (Area under curve). Considering these metrics, we found that the GraphSAGE algorithm performs better than other methods (GCN algorithm and baseline methods) for both link prediction and node classification problems.

In future, we propose to create a more advanced ensemble model of GraphSAGE, GCN and other effective models to improve the current results. We hope to attain better accuracy of the ensemble model so that we get better predictions and correct classifications. We plan to improve feature engineering by creating a robust feature transformation and selection methods to get better node or link embeddings. We propose to perform more extensive hyperparameter tuning to fine-tune the accuracy of the current models. We also plan to improve the data by balancing the samples in each class. With these improvements we hope to achieve better results for our problem.

ACKNOWLEDGEMENT

We would like to thank Rutgers University faculty for the resources and knowledge, Professor Yongfeng Zhang and teaching assistant Zhiqing Hong. We really appreciate all the help and the learning opportunity provided by the faculty. This project would not have been completed without team effort and the help of all the referenced resources.

REFERENCES

[1] Belkin et. al. 2006. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *JMLR* (2006).
[2] Mikolov et. al. 2013. Distributed Representations of Words and Phrases and their Compositionality. *arvix* (2013).
[3] Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations. *arxiv* (2014).
[4] Tang et al. 2015. LINE: Large-scale Information Network Embedding. *arxiv* (2015).
[5] Thomas N. Kipf et al. 2017. Semi-supervised classification with graph neural networks. *ICLR* (2017).
[6] Weston et al. 2012. Deep Learning via Semi-Supervised Embedding. *IGAR* (2012).
[7] Zhu et al. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. *ICMl* (2003).