Evaluating Network Embeddings: Node2Vec vs Spectral Clustering vs GCN

Rahul Palamuttam
Stanford University
450 Serra Mall, Stanford, CA 94305
rpalamut@stanford.edu

William Chen
Stanford University
450 Serra Mall, Stanford, CA 94305
wic006@stanford.edu

Abstract

Node classification on popular social network datasets in the graph setting arises in various real world networks. Being able to label a particular entity in a graph based on its neighboring graph structure and predicting relationships between entities plays an important role in analyzing social networks and content on the web. With the the resurgence of machine learning, supervised learning problems have been shown to accomplish a number of different tasks from classifying animals to determining the model of cars [7]. These methods have been further extended by the resurgence of Deep Learning to accomplish the same tasks such as classifying images with higher accuracy. However, it is noted by Leskovec et al. [2] that prediction tasks on graphs require careful feature engineering. In this paper we adapt work by Kipf et al [4][5] in order to leverage Graph Convolutional Neural Networks as a means of evaluating spectral embeddings in comparison to embeddings generated by the Node2Vec algorithm.

Problem Introduction

With the surge of popularity in various social media services, so too have we experienced a massive influx in the amount of available user data. With so much data floating around, it would be a significant waste for us not to utilize it for the purpose of improving people's lives.

One such avenue through which we may learn about people's behaviors and preferences is via network analysis on social graphs. The underlying networks over which these social media services are built upon are not only rich in information, but they are also subjects of a vast amount of study and research. By applying what we have learned from work that has been done on similar network problems, we hope to gain insight into people's social behaviors.

For this project, we will explore the effectiveness of graph embedding algorithms such as node2vec and computing the top-k eigenvectors in predicting node label assignments. We strive to determine just how much information we are able to get from the structures of a network alone and evaluate this using a classification problem. The algorithms which we will use are the node2vec random walk algorithm [2], a simple spectral embedding technique, and the graph convolutional networks algorithm [4]. We aim to see how well these are able to do on the aforementioned classification problems and in what way we may improve them.

Review of Prior Relevant Work

With the resurgence of Machine Learning and specifically Deep Learning, a whole host of classification and prediction problems have been made amenable to being solved with a high degree of accuracy. However, it is noted by Leskovec et al. [2] that prediction tasks on graphs require careful feature engineering. In this paper we adapt work by Kipf et al [4][5] in order to leverage Graph Convolutional Neural Networks as a means of evaluating node embeddings specifically those generated from the Node2Vec and those generated from taking the top-k eigenvectors of the normalized adjacency matrix. Similar work by grover et al. [2] introduces a novel approach to learning continuous feature representations for nodes in networks.

In addition to GCN, Deep Feature Learning for Graphs has been illustrated in the work by Rossi et al [9] which introduces a framework, DeepGL, for computing a hierarchy of graph representations. Furthermore, there have been efforts made to learn embeddings in a fully unsupervised manner by instead learning subgraph embeddings [6]. Another issue to note is that node2vec does not perform as well with structural equivalence tasks due to the fact that many networks exhibiting strong homophily. In such cases, approaches which try to learn the latent representation of a network will tend to perform poorly [8].

Furthermore, a long standing technique of graph analysis focuses on studying the eigenvalues and eigenvectors of the normalized adjacency matrix of the graph. For example, recent work by [1] looks at leveraging eigenvectors of the Laplacian of a graph to predict climate variability. Eigenvectors of graphs have also been shown to be effective at spectral clustering as shown by [10]. This leads us to believe that the top-k eigenvectors derived from the normalized adjacency matrix of a graph could be an effective embedding technique to represent nodes. That is, each component of the top-k eigenvector corresponds to a k-position in the node feature vector.

Methods

Description of Data Collection Process

We perform our experiments on both the BlogCatalog dataset from Arizona State University and the Cora publication dataset. There was a considerable amount of work put into preprocessing the dataset to be compatible with our Graph Convolution Network input, the node2vec algorithm, and the T-SNE plots.

The preprocessing towards making the BlogCatalog dataset compatible for extracting spectral embeddings involved first extracting the adjacency matrix. We needed to rewrite the csv file storing edges to be parse-able by snap. We then used the snap library to read the graph in and generated an adjacency matrix. From here, we normalized the adjacency matrix and then computed the top 130 eigenvectors. Initially the runtime of the algorithm took more than thirty minutes since we were computing the eigenvectors and eigenvalues of a 10312 x 10312 dense matrix. We instead decided to take advantage of the sparsity of the matrix and utilized the sparse representation to do the eigen computations. This resulted in an improved runtime of a minute.

The BlogCatalog dataset also needed to be compatible with the Graph Convolutional Network input. Towards this goal, we wrote parsers to read the embeddings file that was the output of the node2vec and spectral embeddings algorithms. The corresponding class labels were also transformed into one-hot encoded vectors. Finally, we passed in the entire matrix of node embeddings and node labels as well as a masked array for each of the training, validation, and test sets.

The Cora dataset proved to be significantly more challenging to preprocess. For starters, we needed to generate the edgelist given an adjacency matrix. The fact that the cora dataset was stored as pickled sparse adjacency matrix made it easier to unpickle and preprocess. However, the edgelist itself consisted of nodes from 0 to 2708, but we only had labels from 0 to 1708. We then decided to truncate the edges list (and the corresponding adjacency matrix when generating the spectral embeddings), to the first 1708 nodes.

We also perform our experiments on the Facebook Social Circles dataset from the snap website. The dataset is comprised of 10 subgraphs, each corresponding to a particular "ego user" with the graph containing information about their friends and social circles. Due to limited computational resources and the overwhelming size of the Facebook dataset, we performed experiments on one of these subgraphs, specifically ego user 107. Again due to a lack of computational resources our evaluation of the Facebook dataset is not as extensive as the previous two.

For the Graph Convolutional Network model to work, we had to perform a moderate amount of preprocessing on the data. In the example datasets provided in the GCN github repo, we noticed that all graph's node list were sequential without any missing nodes. For example, if the number of nodes in the graph is 1038, then all of node id's 1 to 1038 are present. This was not the case with the facebook graph, where the total number of node ids did not correspond to the highest value for a node id. For this reason we needed to map each node to a position key and they use the position key for subsequent preprocessing and partitioning of the dataset.

Dataset Analysis and Background

The table below contains some basic information and analyses about the BlogCatalog dataset graph.

Num nodes	10312
Num edges	333983
Avg clustering coefficient	0.4632
Num classes	39
Max node degree	3992

Table 1: BlogCatalog dataset values

We also performed some exploratory analysis on the distribution of node degrees as well as the number of nodes per class in the graph. The results appeared consistent with that of a typical social network where we see few nodes with very large degree, many nodes with low degree, and the degrees exhibiting a large range. Figure 2 on the other hand depicts the distribution of class sizes for the graph. The results are generally consistent with what we expect with few classes having many nodes and the rest having few. The figures are given below.

A brief analysis on the Cora dataset is as follows. Other than the fact that the Cora dataset has much fewer nodes, edges, and classes, we generally observe that it follows similar behaviors as the BlogCatalog dataset. The node degree distribution follows an almost identical curve and the class size distribution is similar in the sense that only one class has a larger size than the rest. It is interesting that the number of classes in the dataset does not have much affect on this behavior. However, we later note that the lack of density in the Cora dataset enables feature embeddings to better capture the structure of the dataset. This is especially apparent in the T-SNE mappings as shown in the evaluations section.

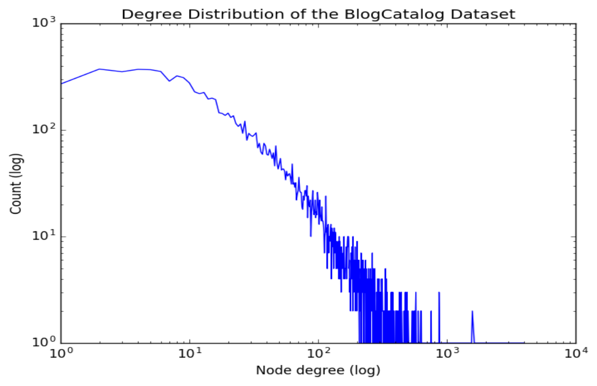


Figure 1: Degree distribution of BlogCatalog dataset

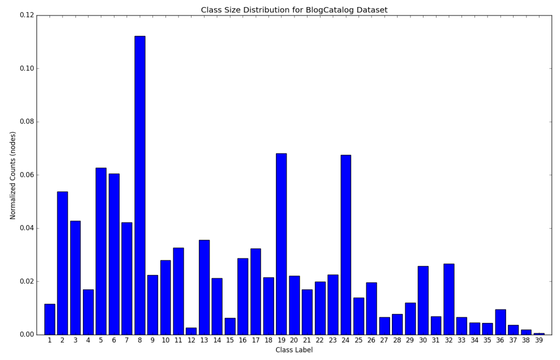


Figure 2: Class size distribution of BlogCatalog dataset

Num nodes	2708
Num edges	5278
Avg clustering coefficient	0.2407
Num groups	7
Max node degree	168

Table 2: Cora dataset values

Finally, while we had also performed some preliminary analyses on the Facebook dataset, we decided not to move forward with it due to the preprocessing overhead and a lack of computational resources. Below we provide some numbers as well as a t-SNE plot using the embeddings generated by the node2vec algorithm. Ideally the plot would give us insight into how well the embeddings are able to represent the circle assignments in the graph. From a quick glance at figure 5, we observe a clear lack of separation between the classes and it becomes immediately apparent that the random walk approach of node2vec is not able to capture the information too well. We are however, interested in whether the learned embeddings from the Graph Convolutional Networks model bootstrapped with the node2vec embeddings can lead to better results.

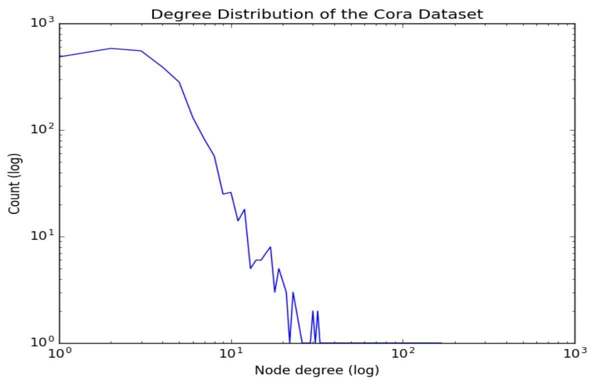


Figure 3: Degree distribution of Cora dataset

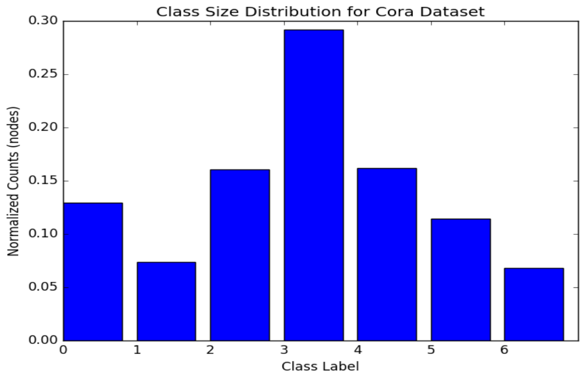


Figure 4: Class size distribution of Cora dataset

Num nodes	1034
Num edges	26749
Avg clustering coefficient	0.6055
Num circles	9
Num nodes in multiple circles	19
Max node degree	253

Table 3: Facebook Social Circles dataset values

Mathematical Background

Node2Vec

We use Node2Vec to obtain node embeddings which we leverage as features to classify the BlogCatalog Dataset, the Cora citation network cataset, and the facebook dataset. We

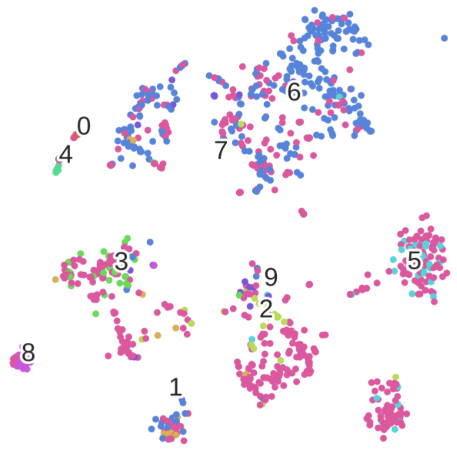


Figure 5: t-SNE plot for node2vec embeddings

specifically determine whether a node belongs to a particular circle of nodes, or class of blog.

In the work by Leskovec et al. [2] feature learning in graphs was characterized as a maximum likelihood problem. They define $N_s(u) \subset U$ as a Networking Neighborhood that is generated through a sampling strategy (BFS, DFS, and Random Walk). The objective function of Node2Vec is given by the following :

$$max_f \sum_{u \in V} logPr(N_s(u)|f(u))$$

Here u is the node and f is the mapping function from nodes to feature representations which will be learned. They make two key assumptions in the paper being
1. The likelihood of observing a neighborhood node is independent of observing any other node. That is :

$$logPr(N_s(u)|f(u)) = \prod_{n \in N_s(u)} Pr(n_i|f(u))$$

2. They assume that a node n_1 and a neighborhood node n_2 have a symmetric effect on their features. Thus they are able to condition the likelihood of a neighborhood pair with the following equation.

$$Pr(n_i|f(u)) = \frac{e^{f(n_i)*f(u)}}{\sum_{v \in V} e^{f(n_i)*f(u)}}$$

Spectral Embeddings

We call the following technique the "Spectral Embedding Technique". However this primarily involves taking the top-k eigenvectors of the normalized adjacency matrix. First we compute the diagonal degree matrix D and normalize the adjacency matrix A.

$$D^{\frac{-1}{2}} A D^{\frac{-1}{2}}$$

We then take the top-k eigenvector where

$$k \in \mathbb{R}$$

$$D^{\frac{-1}{2}} A D^{\frac{-1}{2}} v_k = \lambda_k v_k$$

where λ_k is the k-th largest eigenvalue and v_k is the corresponding eigenvector.

From here we take each eigenvector and for each node n and compose feature vectors u_n where

$$u_{ni} = v_{in}$$

. That is the i'th component of the n'th nodes feature vector corresponds to the n'th component of the i'th eigenvector. Thus in the matrix of eigenvectors, each row-vector is a feature embedding for the corresponding node.

Graph Convolutional Network

We evaluate Node2Vec and the Spectral Embedding technique via node classification with a Graph Convolutional Network. The model takes as input an initial set of features for each node and performs several steps of propagation outwards by following the structure of the graph. The model utilizes a neural network to perform the propagation step while learning a set of weights for each step. Each layer of the neural network corresponds to a step outwards, thus increasing the size of the local neighborhood used to compute each node's embedding. This form of computation is a first-order approximation of the local spectral filters on the graph which improves its computational efficiency. [4]

The model's performance suffers in some instances due to its use of approximations. Notably, when the graph has a large amount of structure, its performance pales in comparison to approaches which use more standard CNN methods. [3]

Each layer of the Approximate Graph Convolutional Network is defined by the following equation:

$$H^{(i+1)} = \sigma(D^{\frac{-1}{2}} A D^{\frac{-1}{2}} H^i W^i)$$

Here hidden layers are given by the $H^{(i+1)}$ term and correspond to the computation of the embeddings by considering the neighbors of nodes at least i steps away. H^0 is essentially the input matrix X which represent the initial features of the node network. Note that A is the identity matrix I added to the adjacency matrix A_{adj}. That is:

$$A = I_n + A_{adj}$$

Adding the identity matrix to the adjacency matrix essentially adds self-edges to all nodes and allows the embeddings to be computed from previous steps. After adding self loops, the adjacency matrix is normalized by multiplying by the inverse of the diagonal degree matrix. This ensures that the feature vectors maintain the same scale between propagation steps.

Formal Description of Algorithms

The Node2vec algorithm is detailed below. Initially the preprocessed modified weights are initialized. All walks are initialized to be empty. And we begin iterating several times. Within each iteration we traverse the list of nodes and begin the node2vecWalk algorithm from that node to generate a walk. The generated walk is then appended to a list of walks. Finally stochastic gradient descent is applied to the list of walks and the result is returned.

Algorithm 1 The Node2Vec algorithm

```
1: procedure LEARN FEATURES(Graph G = (V, E, W), Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q))
2: $\pi$ = PreprocessModifiedWeights(G, p, q)
3: $G' = (V, E, \pi)$
4: Initialize walks to empty
5:     for iter=1 to $r$ do do
6:         for all nodes u in V do do
7:             walk = Node2VecWalk(G', u, l)
8:             Append walk to walks
       $f = StochasticGradientDescent(k, d, walks)$
   return f
```

Results and Findings

We sought to evaluate the effectiveness of the spectral technique and the node2vec technique in both a supervised setting and an unsupervised setting. For the supervised setting we wanted to leverage Graph Convolution Networks as proposed by kipf et al [4] for classification tasks. For the unsupervised setting, we sought a qualitative measure by visualizing the results of running T-SNE on the learned embeddings. The primary question we answer is if embeddings that solely represent graph structure are effective in classification tasks. Towards this goal we utilized a simple but effective technique known as spectral embeddings where we took the first top k eigenvectors corresponding to the top k eigenvalues of the normalized adjacency matrix. We agree that node2vec is a nuanced approach at embedding the structure of graphs and expected classification tasks to do be generally better than the spectral embeddings. Furthermore, we also expected the T-SNE results to be appear fairly separable for node2vec embeddings as opposed to the spectral embeddings. Note that we evaluate the effectiveness of GCN for 2000 epochs and only utilize the spectral and node2vec embeddings, and the adjacency matrix of the respective graphs. We do not utilize the underlying feature vectors of the nodes as GCN already achieves a high classification accuracy of 90 percent when doing so.

In short, isolating the embeddings learned from just the structure of the graph is a more fair assessment.

Cora Citation Network

The Cora Citation is a directed citation network where the nodes represent papers and edges represent that paper A cites paper B. We believed the directed nature of the graph and the inherent temporal nature of citation graphs having more connections with nodes appearing ”around the same time” as opposed to later lends itself to having an interesting structure. However, since the structure of graphs were largely determined by time rather than the type of publication we did not expect to see clear visual distinctions in the T-SNE plot.

Unsupervised Learning Evaluation

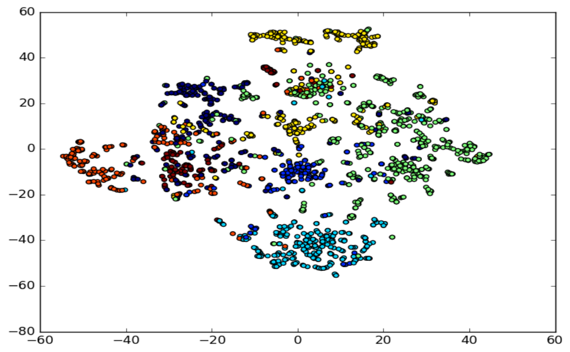


Figure 6: T-SNE plot of the Cora dataset given the node2vec embeddings. T-SNE using the node2vec embeddings after 2000 iterations attained an error of 0.89

In 6 we see the T-SNE plot of the node2vec embeddings and in 7 we see the T-SNE plot of the spectral embeddings.

The T-SNE map constructed from the Node2Vec is significantly better than the map constructed from the spectral clustering embeddings. We clearly see the distinction between different labels as they are physically grouped amongst similar labels. Furthermore there is

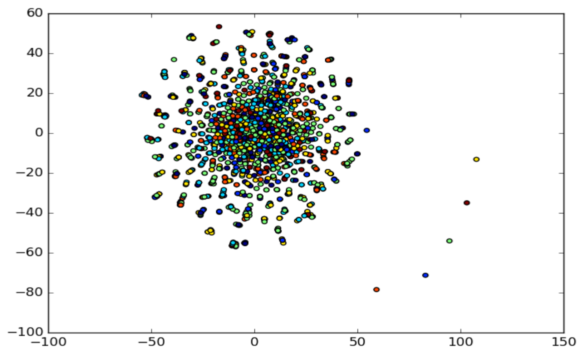


Figure 7: T-SNE plot of the Cora dataset given spectral clustering embeddings. T-SNE using the spectral clustering embeddings after 2000 iterations attained an error of 1.64

a non-negligible degree of separation from each of the groups. The embeddings generated via the spectral clustering method produces a t-SNE plot which shows little to no clustering of the classes. We believe that this is due to us not picking the correct number of eigenvectors when creating the embeddings. We decided to go with the top 130 eigenvectors as our infrastructure limited us to have to use the same embedding dimension size as that of node2vec. However, we believe using only the top 10 eigenvectors would have provided us better results.

Supervised Learning Evaluation

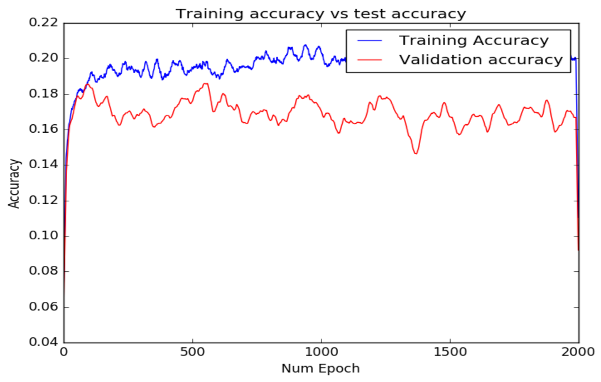


Figure 8: Training vs Test accuracy given the spectral clustering embeddings using GCN

Note that the accuracy of the classification task is low due to the fact that we use no other information other than the structural embeddings and we only train for 2000 epochs. However we use the learning curves to determine the effectiveness of using the embeddings in classification

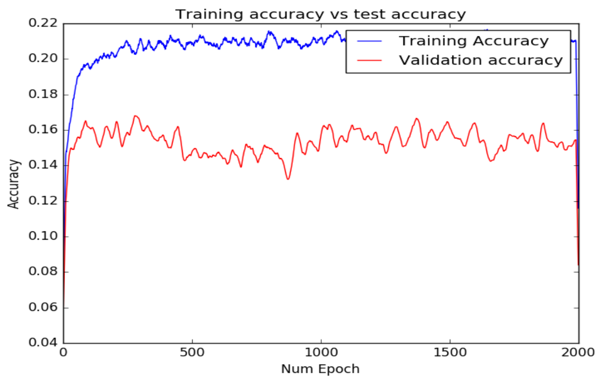


Figure 9: Training vs Test accuracy given the Node2Vec embeddings using GCN

tasks. In Figure 8, which shows the learning curve from the spectral embeddings, we see the that both the training and validation accuracy are fairly similar for each epoch. However, in Figure 9, which shows the learning curve from the node2vec embeddings, we see that the training and validation accuracy have a far greater separation. Both plots display the same overall training accuracy, yet it looks like the more general approach of the using spectral embeddings contributes to an improved quality in training.

Blog Catalog Dataset

The Blog Catalog dataset is the blog directory which manages the bloggers and the blogs they write. The dataset is a network consisting of users being the nodes in the graph and friendship among users being the edges. We were not sure how effective the following graph would be considering the dataset is quite dense consisting of 10312 nodes and 333,983 edges. The classification task here looks at determining which of the 39 groups the users belong to based on the structural embeddings learned from either node2vec or the top 130 eigenvectors.

Unsupervised Learning Evaluation

For the Blog Catalog dataset the dense nature of the graphs detracted from the overall quality of the T-SNE visualizations. Despite this limitation, Figure 10, which denotes the visualization generated from the spectral embeddings, consists of many smaller distinct subgroups that are clearly partitioned. There are no clear distinctions as to which label belongs to which group however. On the other hand Figure 11 contains fewer regions of partitioned points. However, there are distinct large regions that are predominantly one color. The majority of the points appear in the central body of the mapping and really tell us very little about the quality of the embedding. Overall, we see here again that the spec-

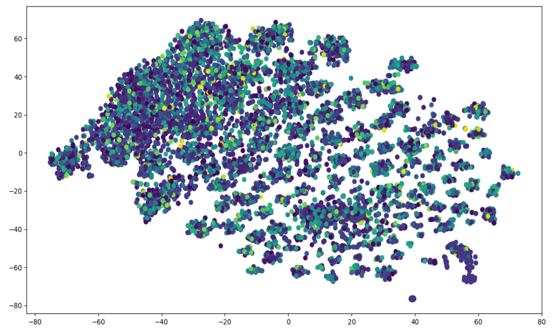


Figure 10: T-SNE plot of the BlogCatalog dataset given the top 130 eigenvector embeddings. T-SNE using the eigenvector embeddings after 1000 iterations attained an error of 1.67

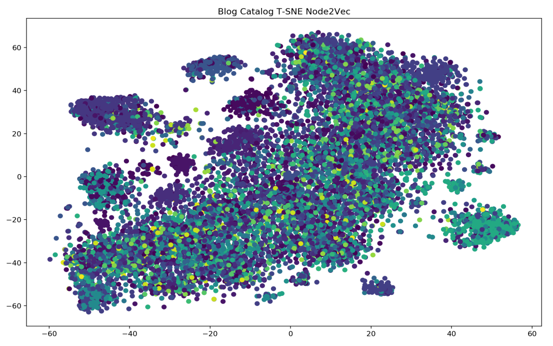


Figure 11: T-SNE plot of the BlogCatalog dataset given the node2vec embeddings. T-SNE using the node2vec embeddings after 1000 iterations attained an error of 2.57

tral embeddings are better at clearly indicating separability in a lower dimension representation amongst nodes in the Blog Catalog dataset when compared to node2vec.

Supervised Learning Evaluation

In Figure 12, which shows the learning curve from the spectral embeddings for the Blog Catalog Dataset, we see that the training accuracy thresholds around 25 percent and the validation accuracy thresholds around 20 percent. However, in Figure 9, which shows the learning curve from the node2vec embeddings, we see that the training and validation accuracy perform relatively poorly achieving 11 and 9 percent accuracy respectively. Here we see that the spectral embeddings are clearly better in the supervised learning setting. It's interesting to note that despite plotting a moving average across the epochs, the variability of the training and validation accuracy is still greater when using the node2vec

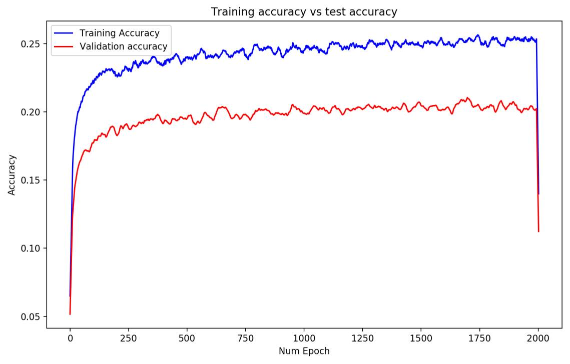


Figure 12: Training vs Test accuracy given the Spectral embeddings using GCN

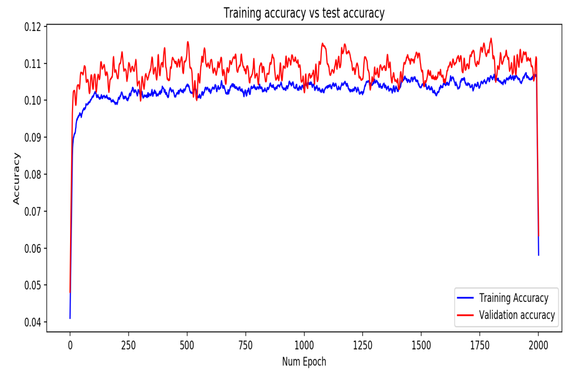


Figure 13: Training vs Test accuracy given the Node2Vec embeddings using GCN

embeddings.

Challenges

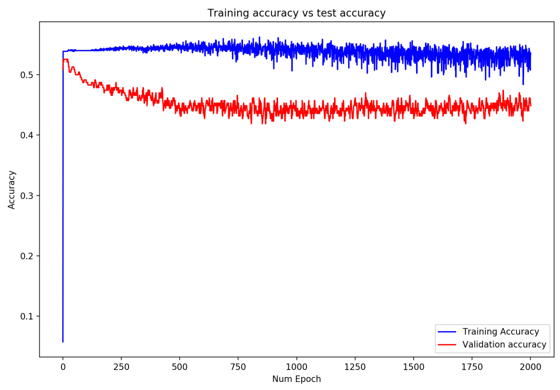
Sparse vs Dense Matrix

This was a rather easy challenge. Graph convolutions are notoriously expensive when the input representation is a dense adjacency matrix. By using the sparse matric CSC format we were able to mitigate this issue. However, it should be noted that while we reduced the size of the input data the dimensionality remains the same. Thus, we think that by expanding both the size and number of hidden layers we could attain better results.

Node Social Circle Imabalance

There is a clear imbalance of class representation in the Facebook dataset, and was our primary challenge in the classification task. More specifically we notice that the majority of nodes belong to social circle 6 or do not belong to any social circle based on Figure 2. We posit a few alternatives to changing this property of the graph but still maintaining the overall structure of the graph. One is to cull excess nodes that belong to class 6. However this would inevitably be affecting the graph structure. Another possible modification is to actually pre-initialize the weights corresponding to nodes belonging to the less represented classes to a higher value. This has the opposite affect of regularization in that we are immediately placing an emphasis on the least represented classes from the onset.

Poor convergence



As shown in the above figure it could entirely be that this is still an ill-proposed problem for the facebook dataset. Our initial results show that we do not do better than classifying 50 percent of the correct social networks. Given that social circle 6 has the majority of nodes, the network could achieve this high percentage by just classifying each example as belonging to social group 6.

Future Work

Our initial attempt at evaluating different node embeddings using GCN was succesfully, all-be-it with many avenues of future improvement. However, the classification tasks in general performed poorly so the embeddings themselves are not too promising. It could also be concluded that graph structure is not a great feature to use in classification tasks by itself. We believe graph structure embeddings should be coupled with contextual embeddings gleaned from the dataset. We first want to improve the result of classifying social circles which nodes belong to. To do this, we plan on doing a number of steps. One is to add guassian noise to the adjacency matrix i.e. add random edges. We want to progressively add random edges so the graph tends to behave as a reydos renyi graph. We think adding randomness to the adjacency matrix will actually enable regions of the convolution to activate at different thresholds. However, this does degrade the sparsity of the graph so we would be incurring a performance loss.
We would also be interested in the effect that attention could play in improving the embeddings learned by GCN. In each propagation step, it makes sense that not all of a node's neighbors should contribute equally in computing the node's embedding. Thus, we brainstormed several ideas such as learning an extra set of learned attention weights (with the same dimension as the adjacency matrix) which we could multiply with the adjacency values to weigh them differently. We also considered integrating an LSTM to learn the relative importance of propagation steps for a given context node. Unfortunately due to restrictions on computational resources and time, we were not able to complete these goals.
Next we want to visualize the feature embeddings learned from the GCN using TSNE to qualitatively assess the learned embeddings. This is primarily a means to have a non-quantitative method comparing to Node2Vec. Questions we want to answer are a) Are the node clusters distinct b) Is there much overlap c) If the cluster visualization is not as effective as that produced by the Node2Vec embeddings then why?
Finally we want to perform the same evaluations on a different social network dataset that also has labeled social groups. We believe the twitter links graph could be a promising next graph to look at. Again it could be that we need to slightly modify the problem due to the large class imbalance. Perhaps if we had a dual class prediction scheme i.e. in addition to social group we also predict another variable such as country of origin. However, this might be difficult because a) the datasets are anonymized and b) the country of origin could strongly dictate social circle.

Contributions

Rahul Palamuttam

Rahul ran all the experiments for the Blog Catalog dataset. This included writing the parser and data preprocessing functions. He also contributed the spectral analysis code which took the top-130 eigenvectors which are used as spectral embedding input to the GCN. Rahul also contributed to the related works section of the paper and the results section of the paper as well as portions of the methodology section, future work and introduction.

William Chen

William ran the experiments for the Cora dataset. This included writing a parser, plots, and t-SNE utility functions. He modified the baseline GCN code to use the embeddings output by either node2vec and spectral clustering while also spending time adapting and improving the GCN model. He also contributed to parts of the results, data preprocessing, and methodology sections in the writeup.

Conclusion

The experience taught us several lessons in evaluating node embeddings using graph convolutional neural networks. The first being that graph convolution computations on large adjacency matrices are quite expensive without GPU resources. Second, we learned that it may have been better to just evaluate node embeddings using simple logistic regression as opposed to a complicated neural network architecture. However, this did give us insight into how structural embeddings could be improved and bootstrapped to perform well in classification algorithms using neural networks. We enjoyed the experience of learning about new algorithms and techniques both from this project and the class 224W overall. We wish to thank Prof. Leskovec and the teaching staff for their dedication and hard work. We hope to use what we learned from this experience in the future, to advancing the frontier of Network Analysis.

References

[1] W. R. Casper and B. Nadiga. A new spectral clustering algorithm. *CoRR*, abs/1710.02756, 2017.

[2] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[3] F. Huszar. How powerful are graph convolutions? *inFERECe*, 2016.

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

[5] T. N. Kipf and M. Welling. Variational graph auto-encoders. 2016.

[6] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *CoRR*, abs/1606.08928, 2016.

[7] D. Needell, R. Saab, and T. Woolf. Simple classification using binary data. *CoRR*, abs/1707.01945, 2017.

[8] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 385–394, New York, NY, USA, 2017. ACM.

[9] N. K. A. Ryan A Rossi, Rong Zhou. Deep feature learning for graphs. *arXiv preprint arXiv:1704.08829*, 2017.

[10] F. Zhao, L. Jiao, H. Liu, X. Gao, and M. Gong. Spectral clustering with eigenvector selection based on entropy ranking. *Neurocomputing*, 73(10-12):1704–1717, 2010.