

Electronic Theses and Dissertations, 2020-

2022

Graph Neural Networks for Improved Interpretability and Efficiency

Patrick Pho
University of Central Florida

 Part of the [Categorical Data Analysis Commons](#), and the [Data Science Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd2020>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Pho, Patrick, "Graph Neural Networks for Improved Interpretability and Efficiency" (2022). *Electronic Theses and Dissertations, 2020-*. 1068.
<https://stars.library.ucf.edu/etd2020/1068>

GRAPH NEURAL NETWORKS FOR IMPROVED INTERPRETABILITY AND EFFICIENCY

by

PATRICK PHO

B.S University of Economics Ho Chi Minh City, 2010

M.S University of Central Florida, 2016

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Statistics & Data Science
in the College of Sciences
at the University of Central Florida
Orlando, Florida

Spring Term
2022

Major Professor: Alexander V. Mantzaris

© 2022 Patrick Pho

ABSTRACT

Attributed graph is a powerful tool to model real-life systems which exist in many domains such as social science, biology, e-commerce, etc. The behaviors of those systems are mostly defined by or dependent on their corresponding network structures. Graph analysis has become an important line of research due to the rapid integration of such systems into every aspect of human life and the profound impact they have on human behaviors. Graph structured data contains a rich amount of information from the network connectivity and the supplementary input features of nodes. Machine learning algorithms or traditional network science tools have limitation in their capability to make use of both network topology and node features. Graph Neural Networks (GNNs) provide an efficient framework combining both sources of information to produce accurate prediction for a wide range of tasks including node classification, link prediction, etc. The exponential growth of graph datasets drives the development of complex GNN models causing concerns about processing time and interpretability of the result. Another issue arises from the cost and limitation of collecting a large amount of annotated data for training deep learning GNN models. Apart from sampling issue, the existence of anomaly entities in the data might degrade the quality of the fitted models. In this dissertation, we propose novel techniques and strategies to overcome the above challenges. First, we present a flexible regularization scheme applied to the Simple Graph Convolution (SGC). The proposed framework inherits fast and efficient properties of SGC while rendering a sparse set of fitted parameter vectors, facilitating the identification of important input features. Next, we examine efficient procedures for collecting training samples and develop indicative measures as well as quantitative guidelines to assist practitioners in choosing the optimal sampling strategy to obtain data. We then improve upon an existing GNN model for the anomaly detection task. Our proposed framework achieves better accuracy and reliability. Lastly, we experiment with adapting the flexible regularization mechanism to link prediction task.

To my parents and my grandmother who always support, encourage, and believe in me.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my PhD adviser, Dr. Alexander V. Mantzaris. Without his support, both emotional and technical guidance, this dissertation would not be possible.

I am deeply grateful to my committee members, Dr. Edgard Maboudou, Dr. Larry Tang, Dr. Gita Reese Sukthankar, and my mentor Dr. Hsin-Hsiung Huang, for their valuable feedback and insights on my research and professional development. My gratitude extends to the department of Statistics and Data Science for funding opportunity to undertake my master and PhD studies.

Special thanks should be given to my friend and colleague Michael Hopwood for many thoughtful discussions about interesting research ideas and for his significant contribution in our publications constituting part of this dissertation. I would like to thank Jianbin Zhu, Yuan Du, and Charles Harrison for sharing their expertise and for collaborating with me on many interesting projects that greatly enhance my technical skills. A sincere thank you to Hayden Hampton and Michael Hopwood for their prompt and diligent proofreading of this dissertation.

I would like to extend my sincere thanks to my friends, lab mates, and colleagues - Md Jibanul Haque Jiban, Haosha Li, and Qing He for the cherished time spent together during my study at UCF. My appreciation also goes to my parents, my grandmother, and my aunts for their encouragement and support throughout my studies.

TABLE OF CONTENTS

LIST OF FIGURES	xi
LIST OF TABLES	xxi
CHAPTER 1: INTRODUCTION	1
Contribution	2
CHAPTER 2: REGULARIZED SIMPLE GRAPH CONVOLUTION (SGC) FOR IMPROVED INTERPRETABILITY OF LARGE DATASETS	5
Introduction	5
Related work	8
Data	10
Circular data	10
Linearly inseparable data	11
Cora dataset	13
Methodology	13
Optimization for regularized SGC	17
Training regularized SGC	23

Tuning hyperparameters	24
Simulation Study	34
Simulation Setting	34
Simulation results	37
Discussion	38
Conclusion	39
 CHAPTER 3: EXPLORING THE VALUE OF NODES WITH MULTICOMMUNITY MEM- BERSHIP FOR CLASSIFICATION WITH GRAPH CONVOLUTIONAL NEU- RAL NETWORKS	40
Introduction	40
Data	43
Methodology	44
Sampling methods	44
Degree	44
PageRank	45
VoteRank	45
Simple Graph Convolution (SGC)	45
Evaluation of Network Topology	47

Results	48
Discussion	59
CHAPTER 4: EXPLORING A LINK BETWEEN NETWORK TOPOLOGY AND AC-	
TIVE LEARNING	61
Introduction	61
Data	62
Real datasets	63
Synthetic dataset	63
Methodology	64
Sampling methods	64
Simple Graph Convolution (SGC)	66
Evaluation of Network Topology	68
Results	69
Conclusions	73
CHAPTER 5: ONE-CLASS GRAPH NEURAL NETWORKS FOR DETECTION OF ANOMA-	
LOUS NODES IN ATTRIBUTED GRAPHS (rh-OCGNN)	75
Introduction	75

Methodology	80
Anomaly detection and hypersphere learning	80
Hypersphere learning on attributed graph data	81
Optimization for OCGNN	84
Training OCGNN	91
rh-OCGNN	92
Optimization for rh-OCGNN	95
Training rh-OCGNN	96
Tuning the radius	97
Data	98
Experiments	99
Results	101
Conclusion	110
CHAPTER 6: LINK PREDICTION WITH SIMPLE GRAPH CONVOLUTION AND REG- ULARIZED SIMPLE GRAPH CONVOLUTION	113
Introduction	113
Methodology	115

SGC	115
Regularized SGC	116
Link prediction	117
Datasets	119
Experiment	120
Results	120
Conclusion	124
CHAPTER 7: CONCLUSION	125
LIST OF REFERENCES	127

LIST OF FIGURES

- Figure 2.1: The application of the proposed methodology to the circular data. Data points are produced about the origin at a fixed radius so that 100 points are equally spaced. The data points have class labels allocated so that there are 50 in each of 2 classes. This is shown in the plot with a line which determines the separation. 11
- Figure 2.2: 15 data points of 2 classes are randomly generated and form 2 distinct clusters residing on the opposite sides of the axis. Sub-figure a) presents the scatter plots of these data points. Sub-figure b) shows the network connections among those data points. In terms of a distance metric between the points, or from using a linear projection, erroneous label assignments can arise from using only the Sub-figure a) data but the incorporation of the network associations shown in Sub-figure b) allows this to be bypassed. The results of the application to the proposed methodology is shown in the Methodology section. 12
- Figure 2.3: The overall framework of regularized SGC 19

Figure 2.4: The above plots show the results of applying the SGC method *with and without regularization upon the synthetic circular data*. In Subfigure a) the dashed line shows the separation line where each side defines the point labels and the solid lines shows the SGC projections learned in Θ , and perfect accuracy is achieved. Sub-figure b) presents multiple subplots of separate independent runs of the proposed regularized SGC methodology where there are different random initializations using gradient descent. Various comparable fittings are found and it can be seen how all the aspects of the regularization upon Θ_R are respected in terms of the magnitude, relative directions between class vectors and the number of components (features) used. 27

Figure 2.5: Applying logistic regression to a set of datapoints where there features are not linearly separable. This is the SGC methodology where $k = 0$ and the network information is not incorporated. 28

Figure 2.6: The results of applying the SGC method on the linearly inseparable data is presented here. Subfigure a) plots the data points and the 2 learned parameter vectors by SGC methodology under different initializations. It can be seen how although the displayed classification vectors within Θ with the network information provide the ability for a lossless prediction. Similarly, for Sub-figure b) it can be seen how a new set of axes for the projection $\mathbf{S}^2\mathbf{X}$ (network and features) the linear separation becomes visible. 30

Figure 2.7: The application of the regularized SGC to the dataset where linear projections are incapable of class separation. Subfigure a) shows the parameter vectors produced by introducing constraints into SGC method on the feature space with original data points and how the classification can then produce perfect accuracy. It can be seen the proposed SGC reduces the effective number of features used in the columns of the matrix Θ_R . Subfigure b) shows the plots for the same set of weight vectors displayed on the projection axes of $\mathbf{S}^2\mathbf{X}$ where each datapoint (node) accumulates feature information from neighbors 2-hops away ($k = 2$). 31

Figure 2.8: The results of applying the regularized SGC with an orthogonal constraint upon the projection achieved by using a change in regularization term L_3 . In Subfigure a) the vector projections within the methodology can be fit so that there is no loss and the orthogonality constraint is satisfied. In Subfigure b) the transformation upon the data with the network information is presented and how within this space the linear projections can separate the classes with the orthogonal vectors. 32

Figure 2.9: The plots showing the results of applying the proposed method of SGC with and without regularization on the features to the dataset of Cora. Subfigure a) shows the heatmap of the class columns of matrix Θ which holds the parameter values for the feature projections of the data \mathbf{X} after the inference with SGC without the regularization. Subfigure b) analogously shows the parameter values but with the inference procedure applying the constraints for the regularization as proposed which produces the shown values of Θ_R . On the bottom right of each plot there are 7 cells with padded values to produce the heatmaps. The columns of the parameter vector correspond to different classes, each shown separately, and the weights applied to each feature belonging to the nodes. It can be seen that the regularization reduces the amount of weighting over the features highlighting key variables. 34

Figure 2.10The distribution of the parameter values inferred for the Cora dataset with the application of the SGC and the regularized SGC. In Subfigure a) the SGC is applied and the histograms of the parameter values for each class in Θ is shown in the plots. Subfigure b) shows the equivalent plots but using the regularized SGC that penalizes the number of features. The majority of the features are around value zero. 35

Figure 2.11Visualizations various network topology from simulated attributed graphs, each with 3 communities (colored). Number of preferential attachment (num_pref) controls the overall amount of connectivity while the inter-graph connectivity probability (inter_p) governs the amount of linkings between subgraphs. . . . 37

Figure 2.12Performance of the SGC and regularized SGC under various network settings. 38

Figure 3.1: High iCV_d networks (Cora, Citeseer) have higher accuracies when sampling nodes from the highest score to lowest score (i.e. 'descending' methods), showing the effectiveness of the node ranking algorithms on a node classification task.	50
Figure 3.2: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skewness for each subset's distribution is annotated above the right graph.	52
Figure 3.3: Low iCV_d networks (Pubmed) have lower accuracy when sampling nodes from the lowest to highest score (i.e. 'ascending' methods), showing the ranking algorithms are inversely beneficial to the node classification task. . .	53
Figure 3.4: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skew for each subset's distribution is annotated above the right graph.	54
Figure 3.5: The performance of this pipeline on the Deezer_Europe social media dataset (plot b) is unusual in that almost all sampling methods are uniformly better than random selection.	55

Figure 3.6: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skew for each subset's distribution is annotated above the right graph. 56

Figure 3.7: Logistic probability (blue line) shows an increasing likelihood of a descending sampling procedure as the coefficient of variance of the degree (iCV_d) increases. Results show a complete separation is defined by iCV_d 59

Figure 4.1: Network visualizations for the 35 generated simulations, each with 3 communities (colored). Traversing along the y-axis shows how these networks topologies change when varying the distance within a communities. Traversing along the x-axis shows how the network topologies changes when varying distance between communities. 65

Figure 4.2: Degree distributions for the 35 generated simulations show that all settings create a relatively scale-free network. Traversing along the y-axis shows how these networks topologies change when varying the distance within a communities. Traversing along the x-axis shows how the network topologies changes when varying distance between communities. 66

Figure 4.3: Simulations report a density of preferred (higher accuracy) sampling direction as a function of network topology (y-axis) and feature similarity (x-axis) shows that the sampling direction is dependent on the network topology. . . . 71

Figure 4.4: High iCV_d networks (Cora, Citeseer) have higher accuracies when sampling nodes from the highest score to lowest score (i.e. 'descending' methods), showing the effectiveness of the node ranking algorithms on a node classification task.	72
Figure 4.5: Low iCV_d networks (Pubmed and amazon-photo) graphs have lower accuracies when sampling nodes from the lowest to highest score (i.e. 'ascending' methods), showing the ranking algorithms are inversely beneficial to the node classification task.	73
Figure 5.1: The overall framework of GAD involving a 2-layer GNN. The GNN maps the original feature space $x \in R^D$ to the node embedding $h^{(2)} \in R^Q$	87
Figure 5.2: The overall framework of our proposed rh-OCGNN for graph anomaly detection (GAD) on attributed graph data. Given an input attributed graph, rh-OCGNN guides GNNs to explore embedding space mapping normal nodes closely within a hypersphere of user-specified radius r	94
Figure 5.3: Average test accuracy, precision, and recall rates (over 10 independent initialization of OCGNN models on benchmark datasets. Albeit achieving promising AUCs (reported in Table 5.2, closer inspection reveals subpar performance of OCGNN models indicating sub-optimal decision threshold, i.e. radius, for anomaly detection task.	103

Figure 5.4: Average radius (over 5 independent initialization) against a range of β (on log scale). The radius is an intrinsic decision threshold to detect anomalous nodes in OCGNN framework. The plot shows sporadic behavior of radius indicating that the mapping learned by OCGNN is not efficient to represent normal class data. 104

Figure 5.5: Average training loss curves (over 5 independent initialization) against a range of β (on log scale). Large losses occur at small values of β indicate the model is unsuccessful in learning representative embedding space for normal class data. 105

Figure 5.6: Average training loss curves (over five independent initialization) against a range of radius (on log scale). As the radius gets larger, less training samples are considered in the training process resulting to larger loss values. 107

Figure 5.7: Tuning process of OCGNN (sub-figure A) and rh-OCGNN (sub-figure B). The plots show average validation accuracy rate curves (over five independent initialization) against a range of hyperparameter (β for OCGNN in sub-figure (a) and r for rh-OCGNN in sub-figure (b)). The proposed rh-OCGNN produces relatively smooth mount-shape validation curves facilitating the tuning process. The optimal radii can be easily selected where the curves reach their peaks. 111

Figure 5.8: Average test accuracy, precision, and recall rates (over 5 independent initialization) of OCGNN (sub-figure A) and rh-OCGNN (sub-figure B). Important hyperparameters of OCGNN (β) and rh-OCGNN (r) are tuned using validation approach where the optimal values are selected to maximize validation accuracy rates. rh-OCGNN renders flexibility to directly tailor the decision threshold, i.e. the radius. Hence, it achieves better performance on all benchmark datasets. 112

Figure 6.1: The plots show the results of applying SGC with and without regularization on the hidden edge embeddings of the Cora dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector’s magnitude over the hidden edge embedding highlighting key variables. 121

Figure 6.2: The plots show the results of applying SGC with and without regularization on the hidden edge embeddings of the Citeseer dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector’s magnitude over the hidden edge embedding highlighting key variables. 122

Figure 6.3: The plots show the results of applying SGC with and without regularization on the the hidden edge embeddings of the Pubmed dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector's magnitude over the hidden edge embedding highlighting key variables. 123

LIST OF TABLES

Table 2.1: Summary statistics of the Cora data set	13
Table 3.1: Dataset statistics and domain-specific information.	44
Table 3.2: The sampling direction is predicted with a high accuracy by studying the skewness of the homogeneity connectivity distribution. Misclassifications are likely caused by a lack of node importance evaluators that are robust to graph topology.	57
Table 4.1: Dataset statistics and domain-specific information.	63
Table 4.2: Optimal sampling results on real datasets	69
Table 5.1: Summary statistics of three citation datasets and the size of train/val/test sets used in this work.	99
Table 5.2: Results of the first experiment where we replicate OCGNN with three popular GNN frameworks - GCN, GAT, and GraphSAGE. Test AUCs (in percentage) averaged over 10 independent initializations and their corresponding standard deviations. Note that the standard deviations are not converted to percentage to keep consistent with [120]. The best results are highlighted in boldface. OC-SAGE outperforms its peers on Citeseer and Pubmed datasets while OC-GAT achieves highest average AUC on Cora dataset.	102

Table 5.3: Test accuracy rates (%) averaged over five independent initialization of rh-OCGNN and OCGNN. The optimal values of β (for OCGNN) and r (for rh-OCGNN) are selected via tuning to maximize the validation accuracy rates. Note that on OCGNN, the radius is not a hyperparameter and is learned during training phase. The best models are highlighted with boldface. Overall, the rh-OCGNN models outperform their OCGNN peers on all benchmark datasets. rh-OC-SGC shines on Cora and Citeseer while rh-OC-GAT achieves best performance on Cora.	109
---	-----

Table 6.1: Summary statistics of three citation datasets and the size of train/val/test sets used in this work.	119
---	-----

CHAPTER 1: INTRODUCTION

Many complex systems in our world can be represented as attributed graphs (networks) where each component is regarded as a node with associated attributes (features) and a collection of edges joining pairs of nodes representing relations between them. For example, in e-Commerce, users and products are considered as nodes and edges connecting between certain users and their purchased items. In chemistry, the molecular structure can be described as a graph whose nodes are atoms and edges are defined by the atomic bonding property. In biology, the information of interaction between proteins of a given organism can be modeled as a graph whose nodes are proteins and edges connect pairs of interacting proteins. Behaviors of such systems are mostly defined by or dependent on their underlying network structure [27, 132, 30].

Research into network based approaches in machine learning and deep learning has been ongoing with the goal of uncovering information hidden within these data structures. The applications of graph analysis have a profound impact on human life. E-commerce platforms can build a learning algorithm to exploit the interaction between users and their purchased items to produce highly accurate recommendations. Physicists use graph-based techniques to analyze and discover new materials, which play an important role in solving many societal and energy challenges [22]. Graph analysis can be categorized into node-level tasks (node classification [85], node regression [74]), edge-level tasks (link prediction [119], edge classification [2]), and graph-level tasks (graph classification [126], graph prediction [125]). A key factor determining the success in these tasks is the effectiveness of the framework in combining the node features and the topological structure of the graph, into representative low-dimensional node embedding (in the form of vectors) [99].

Graph Neural Networks (GNN) belong to a class of neural network operating on the graph domain. It provides a methodological framework for combining node features and structural information

in order to produce expressive node representations (node embeddings) for predictions within a machine learning paradigm. Many variants of GNNs have been proposed and achieved state-of-the-art performance on a variety of tasks [48, 37, 116, 121].

The advent of technology has led to exponential growth of graph structured data with millions of nodes and thousands of features. This causes concerns on the processing time, complexity of the models, and the ability to interpret the results. Training large and complex GNN models also poses a challenge in the modeling pipeline as it often requires large amount of annotated data which might be limited and costly to obtain. Another common obstacle in learning with graph structure data is the existence of anomalies entities (nodes, edges, subgraphs) which do not follow expected patterns. Including these uncommon data in machine learning pipeline might negatively affect the quality of the model.

Contribution

This dissertation is a collection of works addressing the aforementioned challenges in learning with graph structured data. The major contributions are as follows:

- **Regularized Simple Graph Convolution (SGC) For Improved Interpretability Of Large Datasets:**

In chapter 2 and in [79], we investigate the capability of SGC in producing expressive node embedding for node classification task. SGC combines network topology and node features via linear operation, and hence renders a fast and efficient framework for computation. We then explore and propose a flexible regularization mechanism upon SGC to facilitate meaningful interpretation. The regularized SGC is capable of producing sparse weight vectors highlighting important input features for further investigation if needed.

- **Exploring The Value Of Nodes With Multicommunity Membership For Classification With Graph Convolutional Neural Networks:**

Sampling plays an important role in machine learning pipeline, as it provides quality training samples to build the model. However, the process of determining the best sampling method has rarely been studied in the context of graph neural networks. In chapter 3 and in [39], we evaluate the effect of multiple sampling strategies on node classification task using SGC. Our discoveries include an indicative measure of sampling efficiency and a heuristic criterion based on network topology which can be utilized to suggest optimal sampling strategies for practitioners.

- **Exploring A Link Between Network Topology And Active Learning:**

Upon the findings of previous work on the importance of network topology in the selection of training samples, we further study the proposed heuristic measure in chapter 4 and in [38]. We design a comprehensive study to benchmark the proposed measure utilizing a set of synthetic datasets covering a wide variety of network structures and input features. We then derive effective sampling approaches based on the proposed measure to facilitate the task of predicting node label.

- **One-Class Graph Neural Networks For Detection Of Anomalous Nodes in Attributed Graphs (rh-OCGNN):**

In chapter 5, we study another important task in graph analysis, which is the problem of detecting anomalous entities. This problem can be cast as one class classification where all normal nodes are utilized to build a model capable of describing the concept of normality. Anomaly is detected if a new node deviates from the description of the fitted model. We propose a new model (rh-OCGNN) that builds upon a previous GNN but addresses some pitfalls that degrade the model's performance. The proposed model is tested and compared to the previous approach, using various benchmark datasets, showing that it produces better

accuracy and reliability.

- **Link Prediction With Simple Graph Convolution And Regularized Simple Graph Convolution:**

Link prediction is an important task on attributed graph with a wide range of useful applications. Simple link prediction approaches have limitation in their capability to capture network topology and node attributes. Graph Neural Networks (GNNs) provide an efficient framework incorporating node attributes and connectivity to produce informative embedding for many downstream task including link prediction. In chapter 6, we explore how to adapt the flexible regularization mechanism of regularized SGC to the link prediction task with the aim of improving the interpretability of the fitted model.

CHAPTER 2: REGULARIZED SIMPLE GRAPH CONVOLUTION (SGC) FOR IMPROVED INTERPRETABILITY OF LARGE DATASETS

Introduction

Research into network based approaches in machine learning has been ongoing with the growing sizes of networks produced by users on commercial online social networking platforms. The ways in which users interact with each other directly on online social network platforms from independent activities have given rise to networks with billions of users (nodes) [52]. The information provided by users can be used to investigate different phenomena. The interlinking information that produces edges which then can form a network, or series of networks, has been studied in the growing field of network science [70, 10]. Network science offers many tools and approaches to learn and to draw insight about the community structures [35], the centrality distribution [16] of the nodes (users), and provides models for how the networks can grow from an initial set of nodes [42, 6]. There are many applications for these insights such as in targeted advertising [53] where brands seek to have highly central nodes spread advertising content, and another application is in the effort to understand the 'landscape' of political polarization between communities in Twitter [103].

As well as the information of node interlinking that can produce a network (graph), there are user attributes that can provide useful information for improving predictive analytics. A notable technique used by online shopping platforms is *collaborative filtering* [84] which works as a recommendation system to improve the customer's shopping experience by optimizing the product view to those items predicted to be of interest. These new link predictions are based upon past purchases and the historical records of other customers. Here, 'clusters' are formed between groups

of items in a multidimensional space of these choices [105, 115]. This is based on the observation that items are not selected independently of previous purchases and that there is information gained from utilizing the data collected [130]. Another area where predictive analytics has used information to make predictions is in student performance [43]. Logistic regression has been used in situations where a model is required in a decision framework to predict achievements [111]. A key difference between the network-based approach and these approaches is that the information contained in the network and how the links influence the node of concern are excluded from the model, which can play a key role in predicted economic behaviors [41]. In the effort to fuse these sources together for models to incorporate, [100] discusses how user identity linkage across online social networks can be accomplished.

Online social networks have been at the forefront of the interest in networks and approaches which use the interlinking information due to their size and the effect they have on human behaviors [7]. The network topologies of the virtual networks can find applications but they also carry over into the physical world. The techniques have been used in other domains such as examining the centrality in streets of urban spaces [23] which also can be seen as a continuation of the original network/topological graph theoretical formulation of Euler's investigation of the 'Seven Bridges of Konigsberg' problem [28]. As these urban networks fit within networks of urban spaces themselves, multilayer networks [50] are produced that span over the globe allowing an analysis of even global migration patterns [12]. Similarly, it is also possible to consider academic literature as a network with similar properties governing its construction, such as homophily [68]. The nodes in such academic networks are publications and the links are the citations between the articles that provide information of association. There is active research in this field [104] which notes that the key motivation is that researchers can spend considerable amounts of time searching for relevant research to not allocate time on topics already explored with similar approaches. Finding associative research is of importance since it is possible for scholars to be directed in areas already

investigated and waste time, as well as materials, in research such as studies requiring expensive laboratory equipment. Navigating the network to extract relevant research is therefore a key activity in preventing this. The work of [82] discusses how investigators can seek insights from these datasets for the dynamics of growth and the interconnectivity of scientific thought. With the growth of the citation datasets (such as the ones described in the Data section) the concerns on the processing time, complexity of the models and the ability to interpret the results are becoming a key issue.

This then poses key questions about how to process and then reason about the results from large datasets with large variations. Questions about the results even require effort in their interpretation. Work such as [106] looks at the problem from a conceptual perspective on the areas of focus for big data and how the user can interact with the data that are the results of a post-analysis. The work of [19] provides a high-level overview of the tools and approaches available to visually investigate the data, and the results of different methods return. It is possible to include the full set of interpretable outcomes and the full set of relevant data features, but that does produce a challenge for the practitioner to determine which features are of prime interest. A dimensionality reduction approach [83] provides a more effective experience for the practitioner.

Graph Neural Networks (GNNs), [91], provide a methodological framework to combine feature prediction and information from network data in order to produce predictions within a machine learning paradigm. There are many applications ranging from image object positions in a non-euclidean space representation, molecular properties, and citation networks [135]. Therefore this work deals with investigating an even more simple GNN, the Simple Graph Convolution (SGC) [121], which has a simpler methodological definition and a competitive predictive accuracy. In developing a modified SGC, the task of reducing the dimensionality in large datasets with a GNN will be explored. As will be shown in Methodology section, the simplicity of the model allows it to be a basis for extensions that can incorporate constraints such as shrinkage upon the parameters.

This is done in a manner similar to the Lasso regularization procedure [114]. This will allow the large complex datasets to be processed in such a manner as to be interpretable and more accessible in terms of the computations resources required. Models other than the SGC would incorporate more complexity into procedures already complex, making large dataset investigations an increasingly large challenge to apply. The work effectively takes the SGC and extends it so that the model can introduce constraints upon the parameter vectors in such a manner as to allow the model to be more easily interpreted. This allows the parameters for each class to be more sparse and for each class to have less overlap between each other. Altogether this results in a parameter matrix which can more easily be inspected. Our contributions are listed as follows:

- We illustrate the capability of SGC in utilizing network information to separate non-linearly separable data.
- We introduce a flexible regularization scheme built on the basis of SGC that renders a sparse set of fitted parameters highlighting important features for further investigation.

Related work

Convolutional Neural Networks (CNNs) [55] has brought a methodological approach to handle high-dimensional problems more efficiently than other paradigms. As noted in [54] in conjunction with deep learning, CNNs have greatly improved the ability to classify sound and image data. The work of [24] formally introduces how graph-based methods can be used with CNNs. A key contribution of [24] is that the extension of the model to generalize to graphs is founded upon localized graph filters instead of the CNN's localized convolution filter (or kernel). It presents a spectral graph formulation and how filters can be defined with respect to individual nodes in the graph with a certain number of 'hops' distance. These 'hops' are representative of the number of

edges traversed between nodes and are the result of the powers of the adjacency matrix where the number of walks can be calculated [16] (walks are paths that allow node revisits).

An introduction to the motivation from basic principals can be found in [101], where the fundamental analysis operations of signals from regular grids (lattice structures) to more general graphs is developed. The authors in [90] utilize the theory of signals in graphs to show how a shift-invariant convolution filter can be formulated as a polynomial of adjacency matrices. The discussion of how low-pass filters are an underlying principle in the GNN is discussed in [72] which is also described in the work of the SGC. [48] proposes a Graph Convolutional Networks (GCN) by adapting Convolutional Neural Networks (CNNs) for graph-structured data. The GCN learns a graph representation via layer-wise propagation rules that represents localized spectral filters.

The GNN can allow for the augmentation of a users social network and their features to make a more accurate prediction and similarly for an academic paper that the features (keywords or low dimensional representation) with the citation links can more accurately place its relevance. The machine learning framework can introduce large overhead in processing time, especially for large datasets, but fortunately research has shown that simpler GNN models display peak performance [98]. The work of [48] which introduces a semi-supervised approach to GNNs, shows in their appendix B the performance of the methodology with the number of 'layers' employed in the model and how there is an actual degradation of the performance after a few layers. The SGC [121] provides an efficient framework to provide a similar model of data associations as the GCN, but avoids the need for layering in the GCN. The methodology of the SGC allows a single layer of matrix computations with a non-linear activation function. This is similar to the processing steps taken for logistic regression, which can be computed for large datasets very efficiently. Building upon this efficient model allows an investigator to explore further constraints that would be much more computationally demanding with the incorporation of layers.

Data

Three different datasets are employed in order to explore the proposed model, two of them being synthetic and the last one being real datasets which are well explored [67, 87]. The first synthetic data set has two-dimensional features with data points placed in a circle and labels applied on opposite sides of the identity line ($x_1 = x_2$). The other synthetic dataset also has 2 dimensions and is placed in such a way that clustering or a non-network-based model, relying upon distance measures, would incorrectly classify the node labels. More about these 2 datasets is described below.

Circular data

Figure 2.1 shows the synthetic data produced with points allocated along a circle based at the origin. There are 100 points, and 50 of them are allocated to each class placing them on either side of the identity line. A key aspect of this data is that the model will attempt to shrink the feature projections which can incur a penalty on the optimization procedure. The compromise between the error function on the data and the regularization term will require a balance, as a single feature reduces the shrinkage penalty, but the direction for the optimal fit uses both dimensions. This compromise is induced since the optimal projection will be with a vector containing nonnegative parameters for each dimension of equal value in a direction $x_1 = -x_2$, therefore, highlighting the shrinkage of one of the parameters. The network data (the adjacency matrix) is a ring network connecting neighboring nodes.

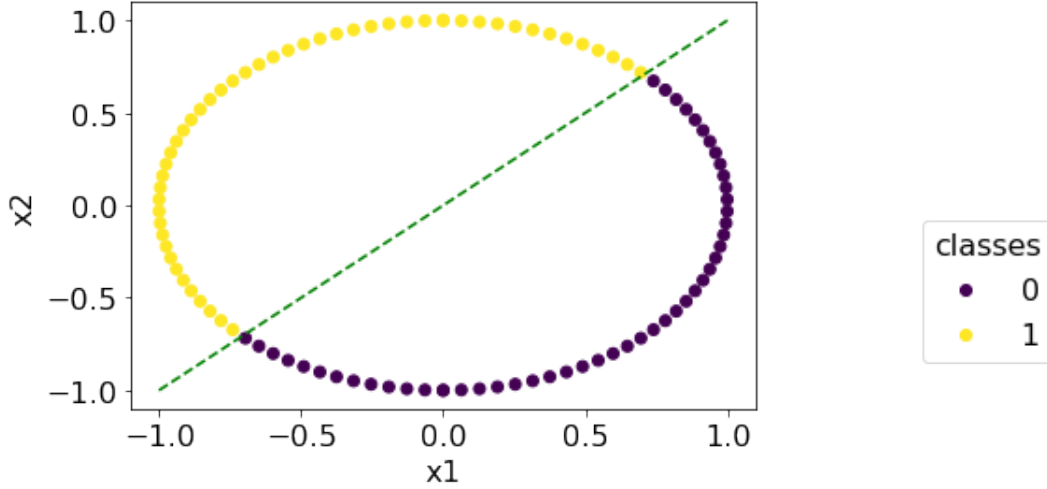


Figure 2.1: The application of the proposed methodology to the circular data. Data points are produced about the origin at a fixed radius so that 100 points are equally spaced. The data points have class labels allocated so that there are 50 in each of 2 classes. This is shown in the plot with a line which determines the separation.

Linearly inseparable data

Figure 2.2 shows 30 synthetically produced data points in 2 dimensions (x_1, x_2) which form 4 distinct clusters. Each class has 15 random generated data points and is separated into 2 clusters across the axis. We produce a nondisjoint network (single component) structure for the data points to be connected with a more dense connectivity set between points of the same label. This production is inline with the concept of modularity in networks [71] where the density of the edges between nodes of the same label is proportionately greater than the density between nodes with different labels. Without the network structure, distance metrics would produce erroneous results, and the introduction of this information increases the accuracy. This allows the linear operations to produce a separation for the class labels.

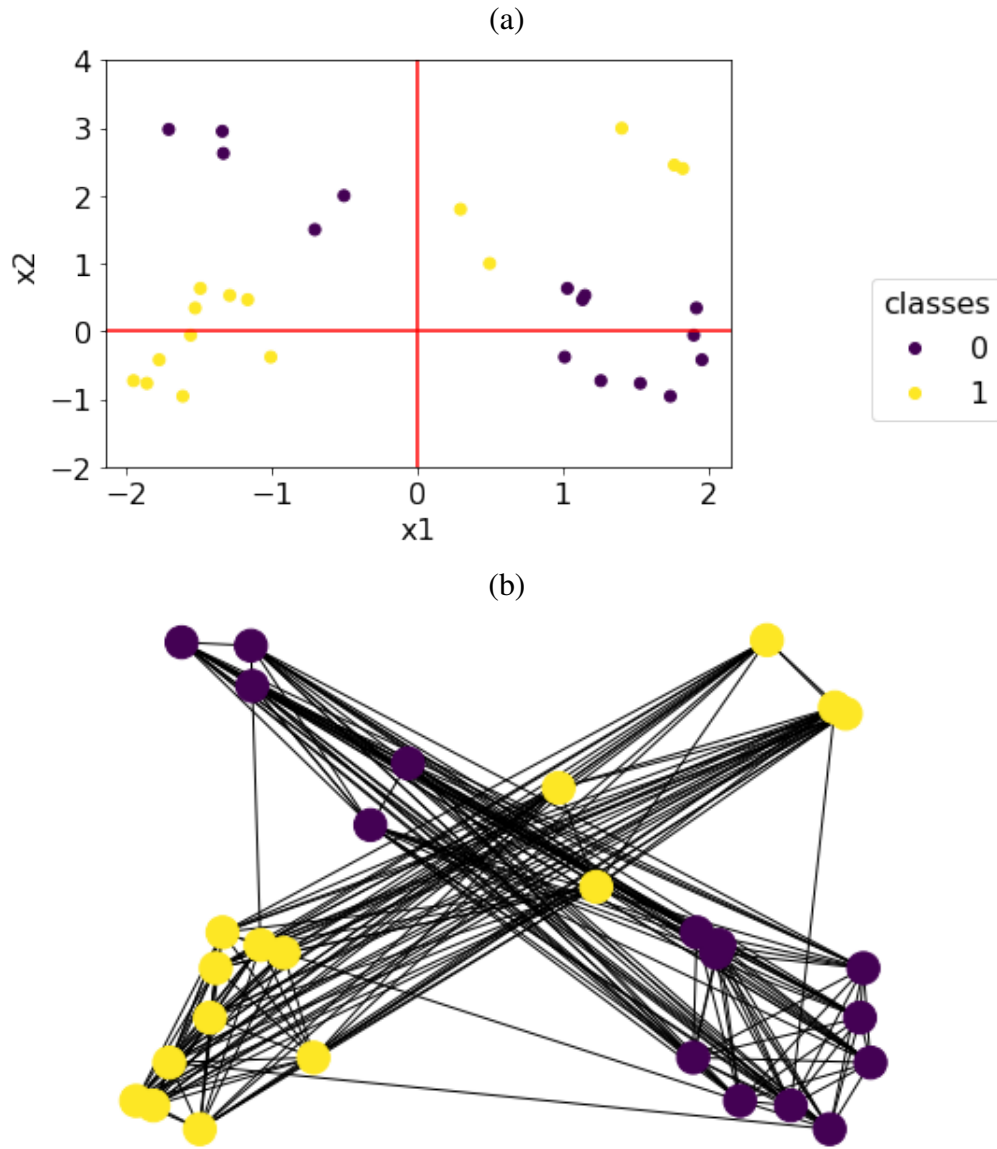


Figure 2.2: 15 data points of 2 classes are randomly generated and form 2 distinct clusters residing on the opposite sides of the axis. Sub-figure a) presents the scatter plots of these data points. Sub-figure b) shows the network connections among those data points. In terms of a distance metric between the points, or from using a linear projection, erroneous label assignments can arise from using only the Sub-figure a) data but the incorporation of the network associations shown in Sub-figure b) allows this to be bypassed. The results of the application to the proposed methodology is shown in the Methodology section.

Cora dataset

Cora [67] is a public citation dataset in which scientific publications are nodes, and the references between them form edges. Each paper’s feature is a binary vector indicating the presence of words in the paper. Papers are categorized by corresponding topics (labels).

Statistics	Value
Number of nodes	2,708
Number of edges	10,556
Number of isolated nodes	0
Number of classes	7
Node feature	1,433
Density	0.002

Table 2.1: Summary statistics of the Cora data set

Methodology

[48] develops Graph Convolutional Networks (GCNs) by adapting Convolutional Neural Networks (CNNs) for graph-structured data, and the work of [121] (proposing the Simple Graph Convolution (SGC)) builds upon it. The SGC removes the nonlinear transitions between the layers in the model. This simplification significantly speeds up processing time, yet still performs on par with GCNs and other state-of-the-art graph neural network models across multiple benchmark graph datasets. The model modification will allow easier interpretability of the parameters fitted by the optimization procedure with the application of a set of constraints. The constraints introduced into the loss function will force the stochastic gradient descent algorithm to find directions that have fewer non-zero values and less overlap for the parameters between the classes. This addresses the problem of how to inspect effectively the matrix of parameters and the vectors of the parameters for each class. This takes inspiration from regularization methods.

We adopt the notations presented in [48] and [121] for the GCN and SGC, respectively. A graph $G = (V; \mathbf{A})$ can be defined as a collection of nodes (vertexes) set $V = (v_1, v_2, \dots, v_N)$ containing N nodes and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ where a_{ij} is the weighted edge between node v_i and v_j ($a_{ij} = 0$ if v_i and v_j are not connected). We define the degree matrix $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_N)$ as a diagonal matrix whose non-diagonal elements are zero, and each diagonal element d_i captures the degree of node v_i and hence $d_i = \sum_j a_{ij}$. There is a feature matrix (also known as the design matrix) $\mathbf{X} \in \mathbb{R}^{N \times D}$ where each row x_i is the feature vector measured at each node of the graph. Each node v_i receives a label from C classes and hence can be coded as one hot vector $y_i \in \{0, 1\}^C$.

The GCNs and SGC add self-loops and normalize the adjacency matrix to get the matrix \mathbf{S} :

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \quad (2.1)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$. This normalization allows successive powers of the matrix to not influence the overall size the projections. The SGC removes non-linear transformation from the k^{th} -layer of the GCN resulting in a linear model of the form:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S} \dots \mathbf{S} \mathbf{X} \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}). \quad (2.2)$$

The SGC classifier is then achieved by collapsing the repetitive multiplication of matrix \mathbf{S} into the k^{th} power matrix \mathbf{S}^K and reparameterizing the successive weight matrices as $\Theta = \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}$:

$$\begin{aligned} \hat{\mathbf{Y}} &= \text{softmax}(\mathbf{S}^K \mathbf{X} \Theta) \\ &= \text{softmax}(\tilde{\mathbf{X}} \Theta) \\ &= \text{softmax}(\mathbf{Z}) \end{aligned} \quad (2.3)$$

where $\bar{\mathbf{X}} = \mathbf{S}^K \mathbf{X}$ is the aggregated input feature and $\mathbf{Z} = \bar{\mathbf{X}} \Theta$ is the weighted input.

The parameter k corresponds to the number of 'hops' which is the number of edge traversals in the network adjacency matrix \mathbf{S} . k can be thought of as accumulating information from a certain number of hops away from a node (as described visually in [121]). If $k = 0$ the methodology becomes equivalent to a logistic regression application which is known to be scalable to large datasets. Since the SGC introduces the matrix \mathbf{S} as linear operation the same scalability applies.

In node classification, we aim to classify the nodes into correct classes. This can be achieved by utilizing SGC $g(\mathbf{X}; \mathbf{A}; \Theta)$ to learn the class distribution $p(\mathbf{y}|\mathbf{X})$, i.e. $p(\mathbf{y}|\mathbf{X}) \approx g(\mathbf{X}; \mathbf{A}; \Theta) = \text{softmax}(\mathbf{S}^K \mathbf{X} \Theta) = \hat{\mathbf{Y}}$. Hence, given K training samples, the set of optimal parameters Θ can be obtained by maximizing the likelihood of data:

$$l(\mathbf{y}|\Theta) = \prod_{i=1}^K \prod_{c=1}^C \hat{p}(y_{ij}|\mathbf{x}_i)^{y_{ic}} = \prod_{i=1}^K \prod_{c=1}^C \hat{y}_{ij}^{y_{ic}} = \prod_{i=1}^K \prod_{c=1}^C [\sigma(z_i)_c]^{y_{ic}} \quad (2.4)$$

where $\mathbf{z}_i = \Theta^T \bar{\mathbf{x}}_i$ is the weighted input feature of sample (node) i , and

$$\sigma(z_i)_c = \left[\frac{\exp(-z_{ic})}{\sum_{c=1}^C \exp(-z_{ic})} \right] = \left[\frac{\exp(-(\Theta_{(:,c)})^T \cdot \bar{\mathbf{x}}_i)}{\sum_{c=1}^C \exp(-(\Theta_{(:,c)})^T \cdot \bar{\mathbf{x}}_i)} \right] \quad (2.5)$$

with $\Theta_{(:,c)}$ is the column c of the weight matrix. Note that this column is the parameter vector corresponding to class c .

Maximizing the problem in 2.4 is equivalent to maximizing the log likelihood:

$$L(\mathbf{y}|\Theta) = \log(l(\mathbf{y}|\Theta)) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \sigma(z_i)_c \quad (2.6)$$

Formally, the SGC model attempts to minimize the objective loss function:

$$\mathcal{L}(\Theta) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \sigma(z_i)_c = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-(\Theta_{(:,c)})^T \cdot \bar{\mathbf{x}}_i)}{\sum_{c=1}^C \exp(-(\Theta_{(:,c)})^T \cdot \bar{\mathbf{x}}_i)} \right] \quad (2.7)$$

As motivated in the Introduction section, the SGC shows how an efficient formulation of GNN can be derived, it does not provide as well the ability to reduce the feature set. To reduce the number of parameter values, we introduce a flexible set of constraints as shrinkage operators in the loss of Eqn 2.7:

$$\begin{aligned} \mathcal{L}_R(\Theta) = \mathcal{L} + L_1 \times \sum_{c \in C} \left(\sum_{d=1}^D |\Theta_{R(\cdot,c)}|^4 \right)^{(-1)} + L_2 \times \sum_{c \in C} \|\Theta_{R(\cdot,c)}\|_2 + \\ L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} \left(|\Theta_{R(\cdot,c_1)}^T \cdot \Theta_{R(\cdot,c_2)}| : c_1 \prec c_2 \right) \right) \end{aligned} \quad (2.8)$$

The first component of \mathcal{L}_R is the loss from SGC being \mathcal{L} . Next, L_1 is the shrinkage term for penalizing the number of parameters by reducing the penalization with a larger skew in the number of elements in the columns of Θ_R . The term $|\Theta_{R(\cdot,c)}|^4$ denotes the normalized vector for each class projection in the parameter matrix (which are columns) and that each element is raised to the power of 4. The term L_2 is the total magnitude of the parameter vector so that the distribution of the terms is not influential but only the norm result. The term L_3 penalizes the projection of class labels that have large overlaps, so that the vectors will be orthogonal or depending on the value of L_3 to support opposing directions. The parameters of the fitted model using the shrinkage in the loss will be referred to as Θ_R . To impose an orthogonality constraint between the projection vectors,

the term L_3 can be slightly modified:

$$\begin{aligned} \mathcal{L}_R = \mathcal{L} + L_1 \times \sum_{c \in C} \left(\sum_{d=1}^D |\Theta_{R(\cdot, c)}|^4 \right)^{(-1)} + L_2 \times \sum_{c \in C} \|\Theta_{R(\cdot, c)}\|_2 + \\ L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} \left(|\Theta_{R(\cdot, c_1)}^T| \cdot |\Theta_{R(\cdot, c_2)}| : c_1 \prec c_2 \right)^2 \right). \end{aligned} \quad (2.9)$$

This methodology therefore delivers a formulation which is based upon an approach with layers as other 'deep learning' frameworks provide, but without the computational burdens that come along with it. Therefore, the simplified model implementation is capable of running on a personal computer with Pytorch [45].

Optimization for regularized SGC

In this section, we present the approach to minimize the loss in Eq. 2.8. The overall regularized SGC framework is shown in Figure 2.3.

For an input attributed graph data consisting of a node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the SGC model first aggregates neighborhood features and updates the input feature of node v_i as:

$$\bar{\mathbf{x}}_i \leftarrow \frac{1}{d_i + 1} \mathbf{x}_i + \sum_{j=1}^n \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}} \mathbf{x}_j \quad (2.10)$$

This can be expressed compactly as $\bar{\mathbf{X}} = \mathbf{S}^k \mathbf{X}$ where \mathbf{S} is the adjusted adjacency matrix with added self-loops.

Then, the weighted input is computed by linearly transforming the input $\bar{\mathbf{x}}$ using a learnable weight matrix $\Theta \in \mathbb{R}^{D \times C}$:

$$\mathbf{z} = \Theta^T \times \bar{\mathbf{x}} \quad (2.11)$$

The output of the model $\mathbf{h} \in \mathbb{R}^C$, is produced by applying a softmax function $\sigma(\cdot)$ to the weighted input:

$$\mathbf{h} = \sigma(\mathbf{z}) \quad (2.12)$$

where

$$\sigma(\mathbf{z})_i = \left[\frac{\exp(-z_c)}{\sum_{c=1}^C \exp(-z_c)} \right] \quad (2.13)$$

Then, with K training samples, the loss function can be constructed as:

$$\begin{aligned} \mathcal{L}_R(\Theta) = \mathcal{L}(\Theta) + L_1 \times \sum_{c \in C} \left(\sum_{d=1}^D |\Theta_{R(\cdot, c)}|^4 \right)^{(-1)} + \\ L_2 \times \sum_{c \in C} \|\Theta_{R(\cdot, c)}\|_2 + L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} \left(|\Theta_{R(\cdot, c_1)}^T| \cdot |\Theta_{R(\cdot, c_2)}| : c_1 \prec c_2 \right) \right) \end{aligned} \quad (2.14)$$

where

$$\mathcal{L}(\Theta) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-(\Theta_{(\cdot, c)})^T \cdot \bar{\mathbf{x}}_i)}{\prod_{c=1}^C \exp(-(\Theta_{(\cdot, c)})^T \cdot \bar{\mathbf{x}}_i)} \right] \quad (2.15)$$

To simplify the notation for subsequent derivation, let $\theta_c = (\Theta_{(\cdot, c)})^T$. Then, we can rewrite the above loss function as:

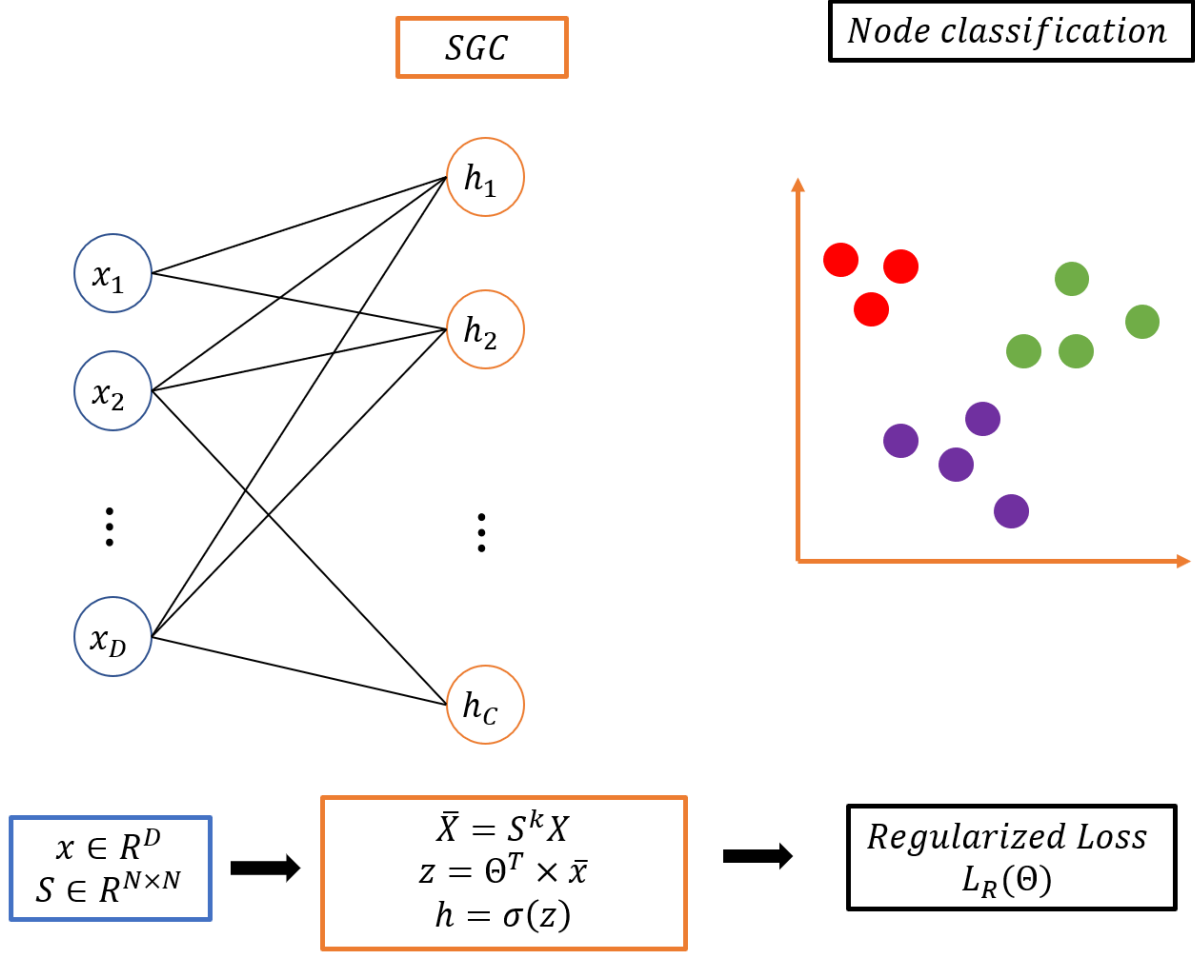


Figure 2.3: The overall framework of regularized SGC

$$\begin{aligned}
 \mathcal{L}_R(\Theta) = \mathcal{L}(\Theta) + L_1 \times \sum_{c \in C} \left(|\theta_c|^4 \cdot \vec{1} \right)^{(-1)} + \\
 L_2 \times \sum_{c \in C} \|\theta_c\|_2 + L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} (|\theta_{c_1}^T| \cdot |\theta_{c_2}| : c_1 \prec c_2) \right) \quad (2.16)
 \end{aligned}$$

where:

$$\mathcal{L}(\Theta) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-\theta_c \cdot \bar{\mathbf{x}}_i)}{\sum_{c=1}^C \exp(-\theta_c \cdot \bar{\mathbf{x}}_i)} \right] \quad (2.17)$$

To minimize the loss in Eq. 2.16, we utilize gradient descent algorithm which requires the gradient of the parameters Θ .

Let $\mathcal{L}_i = \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-\theta_c \cdot \bar{\mathbf{x}}_i)}{\sum_{c=1}^C \exp(-\theta_c \cdot \bar{\mathbf{x}}_i)} \right] = \sum_{c=1}^C y_{ic} \times [-\theta_c \cdot \bar{\mathbf{x}}_i - \log(\sum_{c=1}^C \exp(-\theta_c \cdot \bar{\mathbf{x}}_i))]$ be the loss of a training sample i .

We compute the gradient w.r.t. Θ using the following procedure. For each $c = 1, 2, \dots, C$:

1. compute the gradient of L_i w.r.t. θ_c :

$$\frac{\partial \mathcal{L}_i}{\partial \theta_c} = \left[-y_{ic} - \sum_{c=1}^C y_{ic} \times \left[\frac{\exp(-\theta_c \cdot \bar{\mathbf{x}}_i)}{\sum_{c=1}^C \exp(-\theta_c \cdot \bar{\mathbf{x}}_i)} \right] \right] \times \bar{\mathbf{x}}_i \quad (2.18)$$

2. Compute the gradient of the L_1 term w.r.t. θ_c :

$$\begin{aligned} \frac{\partial}{\partial \theta_c} \left(L_1 \times \sum_{c \in C} (|\theta_c|^4 \cdot \vec{1})^{(-1)} \right) &= \frac{\partial}{\partial \theta_c} \left(L_1 \times (|\theta_c|^4 \cdot \vec{1})^{(-1)} \right) \\ &= -L_1 \cdot (|\theta_c|^4 \cdot \vec{1})^{(-2)} \times \frac{\partial}{\partial \theta_c} (|\theta_c|^4 \cdot \vec{1}) \\ &= -L_1 \cdot (|\theta_c|^4 \cdot \vec{1})^{(-2)} \times (|\theta_c|^4) \cdot \frac{\partial}{\partial \theta_c} (|\theta_c|^4) \\ &= -L_1 \cdot (|\theta_c|^4 \cdot \vec{1})^{(-2)} \times (|\theta_c|^4) \cdot \Delta_c \end{aligned} \quad (2.19)$$

where:

$$\Delta_c = \frac{\partial}{\partial \theta_c} (|\theta_c|^4) = \begin{bmatrix} \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{c1}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{c1}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{c1}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) \\ \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{c2}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{c2}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{c2}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{cD}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{cD}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{cD}^4}{\sqrt{\sum_d \theta_{cd}^2}} \right) \end{bmatrix} \quad (2.20)$$

and:

$$\frac{\partial \frac{\theta_{cj}^4}{\sqrt{\sum_d \theta_{cd}^2}}}{\partial \theta_{ci}} = \begin{cases} 4\theta_{ci}^3 \sqrt{\sum_d \theta_{cd}^2} - (\sum_d \theta_{cd}^2)^{3/2} \cdot \theta_{ci}^5 & \text{if } i = j \\ (\sum_d \theta_{cd}^2)^{3/2} \cdot \theta_{cj}^4 \cdot \theta_{ci} & \text{else} \end{cases} \quad (2.21)$$

3. Compute the gradient of L_2 term w.r.t. θ_c :

$$\frac{\partial}{\partial \theta_c} \left(L_2 \times \sum_{c \in C} \|\theta_c\|_2 \right) = \frac{\partial}{\partial \theta_c} (L_2 \times \|\theta_c\|_2) = L_2 \times \frac{\theta_c}{\|\theta_c\|_2} \quad (2.22)$$

4. Compute the gradient of the L_3 term w.r.t. θ_c :

$$\begin{aligned} \frac{\partial}{\partial \theta_c} \left(L_3 \times \sum_{c_1 \in C} \sum_{c_2 \in C} (|\theta_{c_1}^T| \cdot |\theta_{c_2}| : c_1 \prec c_2) \right) &= L_3 \times \frac{\partial}{\partial \theta_c} \left(\sum_j (|\theta_c| \cdot |\theta_j^T| : j < c) \right) \\ &= L_3 \times \left(\sum_{j, j < c} |\theta_j| \right) \frac{\partial |\theta_c|}{\partial \theta_c} \\ &= L_3 \times \left(\sum_{j, j < c} |\theta_j| \right) \Omega_c \end{aligned} \quad (2.23)$$

where:

$$\Omega_c = \frac{\partial |\theta_c|}{\partial \theta_c} = \begin{bmatrix} \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{c1}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{c1}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{c1}}{\sqrt{\sum_d \theta_{cd}^2}} \right) \\ \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{c2}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{c2}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{c2}}{\sqrt{\sum_d \theta_{cd}^2}} \right) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \theta_{c1}} \left(\frac{\theta_{cD}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \frac{\partial}{\partial \theta_{c2}} \left(\frac{\theta_{cD}}{\sqrt{\sum_d \theta_{cd}^2}} \right) & \cdots & \frac{\partial}{\partial \theta_{cD}} \left(\frac{\theta_{cD}}{\sqrt{\sum_d \theta_{cd}^2}} \right) \end{bmatrix} \quad (2.24)$$

and:

$$\frac{\partial \frac{\theta_{cj}}{\sqrt{\sum_d \theta_{cd}^2}}}{\partial \theta_{ci}} = \begin{cases} \sqrt{\sum_d \theta_{cd}^2} - (\sum_d \theta_{cd}^2)^{3/2} \cdot \theta_{ci}^2 & \text{if } i = j \\ (\sum_d \theta_{cd}^2)^{3/2} \cdot \theta_{cj} \cdot \theta_{ci} & \text{else} \end{cases} \quad (2.25)$$

Then, the loss gradient in Eq. 2.16 w.r.t. θ_c can be achieved as:

$$\frac{\partial \mathcal{L}_R}{\partial \theta_c} = \frac{1}{K} \sum_{i=1}^K \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(1)}} - L_1 \cdot \left(|\theta_c|^4 \cdot \vec{1} \right)^{(-2)} \cdot (|\theta_c|^4) \cdot \Delta_c + L_2 \cdot \frac{\theta_c}{\|\theta_c\|_2} + L_3 \cdot \left(\sum_{j,j < c} |\theta_j| \right) \Omega_c \quad (2.26)$$

Now, to minimize the loss, we iteratively update the parameters Θ with gradient descent algorithm.

In each iteration, we simultaneously update θ_c for $c = 1, 2, \dots, C$ as follows:

$$\theta_c^{(t+1)} \leftarrow \theta_c^{(t)} - \eta_{\Theta} \frac{\partial \mathcal{L}_R}{\partial \theta_c} \quad (2.27)$$

where η_{Θ} is step size of the update.

Training regularized SGC

The training procedure for regularized SGC can be summarized in Algorithm 1. At the outset, the model aggregates node neighboring features to update the node feature input $\bar{\mathbf{X}} = \mathbf{S}^k \mathbf{X}$. The set of weights Θ is initialized and is used to produce initial weighted input via linear operation $\mathbf{Z} = \bar{\mathbf{X}} \Theta$. Then, the node output matrix is computed using softmax function $\mathbf{H} = \text{softmax}(\mathbf{Z})$. The loss is derived as in Eq. 2.8. The framework can optimize the loss by updating the network weights Θ using stochastic gradient descent. In this work, we utilize PyTorch [77] to perform updating the network's weights.

Algorithm 1 Training regularized SGC model

Input: Attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$, training nodes \mathbf{V}_{tr} , number of hops $k \in [0, \infty)$, penalty terms L_1, L_2, L_3

Output: Weight $\Theta = [\theta_1^T, \theta_2^T, \dots, \theta_C^T]$

- 1: Initialize $\Theta^{[i=0]}$
- 2: Aggregate input $\bar{\mathbf{X}} = \mathbf{S}^k \mathbf{X}$
- 3: **while** $i \leq \text{num_iter}$ **do**
- 4: Compute the weighted input: $\mathbf{Z}^{[i]} = \bar{\mathbf{X}} \Theta^{[i=0]}$
- 5: Compute node output: $\mathbf{H}^{[i]} = \text{softmax}(\mathbf{Z}^{[i]})$
- 6: Compute the loss:

$$\begin{aligned} \mathcal{L}_R(\Theta) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-(\Theta_{(.,c)})^T \cdot \bar{\mathbf{x}}_i)}{\prod_{c=1}^C \exp(-(\Theta_{(.,c)})^T \cdot \bar{\mathbf{x}}_i)} \right] + L_1 \times \sum_{c \in C} \left(|\theta_c|^4 \cdot \vec{1} \right)^{(-1)} + \\ L_2 \times \sum_{c \in C} \|\theta_c\|_2 + L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} (|\theta_{c_1}^T| \cdot |\theta_{c_2}| : c_1 \prec c_2) \right) \end{aligned}$$

- 7: Update $\Theta^{[i+1]}$ using gradient descent algorithms
 - 8: $i \leftarrow i + 1$
 - 9: **end while**
-

Tuning hyperparameters

Our regularization scheme allows users considerable leeway in specifying the desired values of penalty terms L_1, L_2, L_3 based on their domain knowledge and applications. If users only wish to obtain a sparse set of parameters, they can apply only the L_1 term. If they believe the parameter vectors are not only sparse but also have the least amount of overlapping, then a combination of L_1 and L_3 terms together might produce a desirable solution.

In some applications, without prior knowledge or domain requirement for the penalty terms L_1, L_2 and L_3 ; users have the option to perform hyperparameters optimization to achieve the best performance. Here, we describe a grid search process to tune L_1, L_2, L_3 in Algorithm 2. First, the labeled samples are partitioned into a training set \mathbf{V}_{tr} and a validation set \mathbf{V}_{val} . Then we choose a performance metric \mathcal{M} to evaluate the capability of the model and set of candidate values $arr_{L_1} = \{a_1, a_2, \dots, a_J\}$, $arr_{L_2} = \{b_1, b_2, \dots, b_K\}$, and $arr_{L_3} = \{c_1, c_2, \dots, c_L\}$ of interest. For each combination of $L_1 = a_j, L_2 = b_k, L_3 = c_l$ belonging to the candidate sets, we train the model on the training set and store its performance on the validation set $\mathcal{M}_{(j,k,l)}$. Depending on the application, the optimal values L_1^*, L_2^*, L_3^* are chosen as a combination that maximizes or minimizes the corresponding set of performance metrics:

$$\begin{aligned}
 L_1^*, L_2^*, L_3^* &= \min_{a_j, b_l, c_k} \{ \mathcal{M}_{(a_1, b_1, c_1)}, \mathcal{M}_{(a_1, b_1, c_2)}, \dots, \mathcal{M}_{(a_J, b_K, c_L)} \} \\
 &OR \\
 L_1^*, L_2^*, L_3^* &= \max_{a_j, b_l, c_k} \{ \mathcal{M}_{(a_1, b_1, c_1)}, \mathcal{M}_{(a_1, b_1, c_2)}, \dots, \mathcal{M}_{(a_J, b_K, c_L)} \}
 \end{aligned} \tag{2.28}$$

Algorithm 2 Tuning hyperparameters L_1, L_2, L_3 for regularized SGC

Input: Attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$, training nodes \mathbf{V}_{tr} , validation nodes \mathbf{V}_{val} , arrays of $arr_{L_1} = \{a_1, \dots, a_J\}$, $arr_{L_2} = \{b_1, \dots, b_K\}$, and $arr_{L_3} = \{c_1, \dots, c_L\}$

Output: Optimal L_1^*, L_2^*, L_3^* and Weight Θ^*

```
1: for  $a_j$  in  $arr_{L_1}$  do
2:   for  $b_k$  in  $arr_{L_2}$  do
3:     for  $c_l$  in  $arr_{L_3}$  do
4:       Assign the penalty terms  $L_1 = a_j, L_2 = b_k, L_3 = c_l$ 
5:       Initialize  $\Theta^{[i=0]}$ 
6:       Aggregate input  $\tilde{\mathbf{X}} = \mathbf{S}^k \mathbf{X}$ 
7:       while  $i \leq \text{num\_iter}$  do
8:         Compute the weighted input:  $\mathbf{Z}^{[i]} = \tilde{\mathbf{X}} \Theta^{[i=0]}$ 
9:         Compute node output:  $H^{[i]} = \text{softmax}(\mathbf{Z}^{[i]})$ 
10:        Compute the loss:
```

$$\mathcal{L}_R(\Theta) = \sum_{i=1}^K \sum_{c=1}^C y_{ic} \log \left[\frac{\exp(-(\Theta_{(.,c)})^T \cdot \tilde{\mathbf{x}}_i)}{\prod_{c=1}^C \exp(-(\Theta_{(.,c)})^T \cdot \tilde{\mathbf{x}}_i)} \right] + L_1 \times \sum_{c \in C} \left(|\theta_c|^4 \cdot \vec{1} \right)^{(-1)} +$$
$$L_2 \times \sum_{c \in C} \|\theta_c\|_2 + L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} (|\theta_{c_1}^T| \cdot |\theta_{c_2}| : c_1 \prec c_2) \right)$$

```
11:        Update  $\Theta^{[i+1]}$  using gradient descent algorithms
12:         $i \leftarrow i + 1$ 
13:      end while
14:      Check model's performance on validation set and store  $\mathcal{M}_{(j,k,l)}$ 
15:    end for
16:  end for
17: end for
18:  $L_1^*, L_2^*, L_3^* = \min_{(a_j, b_l, c_k)} \{ \mathcal{M}_{(a_1, b_1, c_1)}, \dots, \mathcal{M}_{(a_J, b_K, c_L)} \}$  OR  $\max_{(a_j, b_l, c_k)} \{ \mathcal{M}_{(a_1, b_1, c_1)}, \dots, \mathcal{M}_{(a_J, b_K, c_L)} \}$ 
```

Results

Here, we present the results of applying the proposed methodology to the datasets described in Data section. The synthetic circularly placed datapoints with labels allocated on the sides of the identity line of 2 dimensions, described in the subsection Circular data of the Data section. The synthetic datapoints placed along 2 dimensions without a linear separation of the labels based upon a distance metric but possible with the network information is described in the subsection

'Linearly inseparable data' and the results for it are shown in the subsection of the Results section, 'Synthetic linearly inseparable data'. The results of the application to the real dataset of [67] (Cora citation dataset) are shown in the subsection of the Results section, 'Application to the Cora dataset'. The methods of logistic regression, SGC and the regularized SGC are applied and the results are compared revealing that the fitted parameter vectors for each class have less overlap between themselves so that their characteristics for the classes can be more effectively interpreted.

Synthetic circular data

The results of applying the SGC methodology with and without regularization to synthetic circular data are shown in Figure 2.4. The points in the dataset have 2 features x_1 and x_2 . In Subfigure a) the SGC model produces a perfect accuracy with 2 parameters used the projection vectors pointing to the proper direction of each class. The regularized SGC returns different solutions due to the regularization in the parameter matrix Θ . Subfigure b) shows the regularized parameter vectors under different initializations of the learning algorithm which applies constraints. These constraints reduce the number of parameters used, the size of the vectors as a norm, and the direction between the vectors to be more informative. It can be seen how different random initializations produce different loss values and accuracy depending on the local optima reached. These different stable points do show that the shrinkage factors are affecting the vectors for each class in Θ .

A separation based upon the identity line for the 2 dimensions represents a situation where there is equal weight upon all the features of the data and the inference scheme must make a choice in the penalization. The choice results in a decrease in the loss of accuracy in order to decrease the penalization from the regularization from the 3 components calculated from the projections; L_1 , L_2 and L_3 as discussed in Methodology section. The variation shows that the model is able to explore a wide range of vectors for the matrix columns of Θ_R . From that the choice with the largest

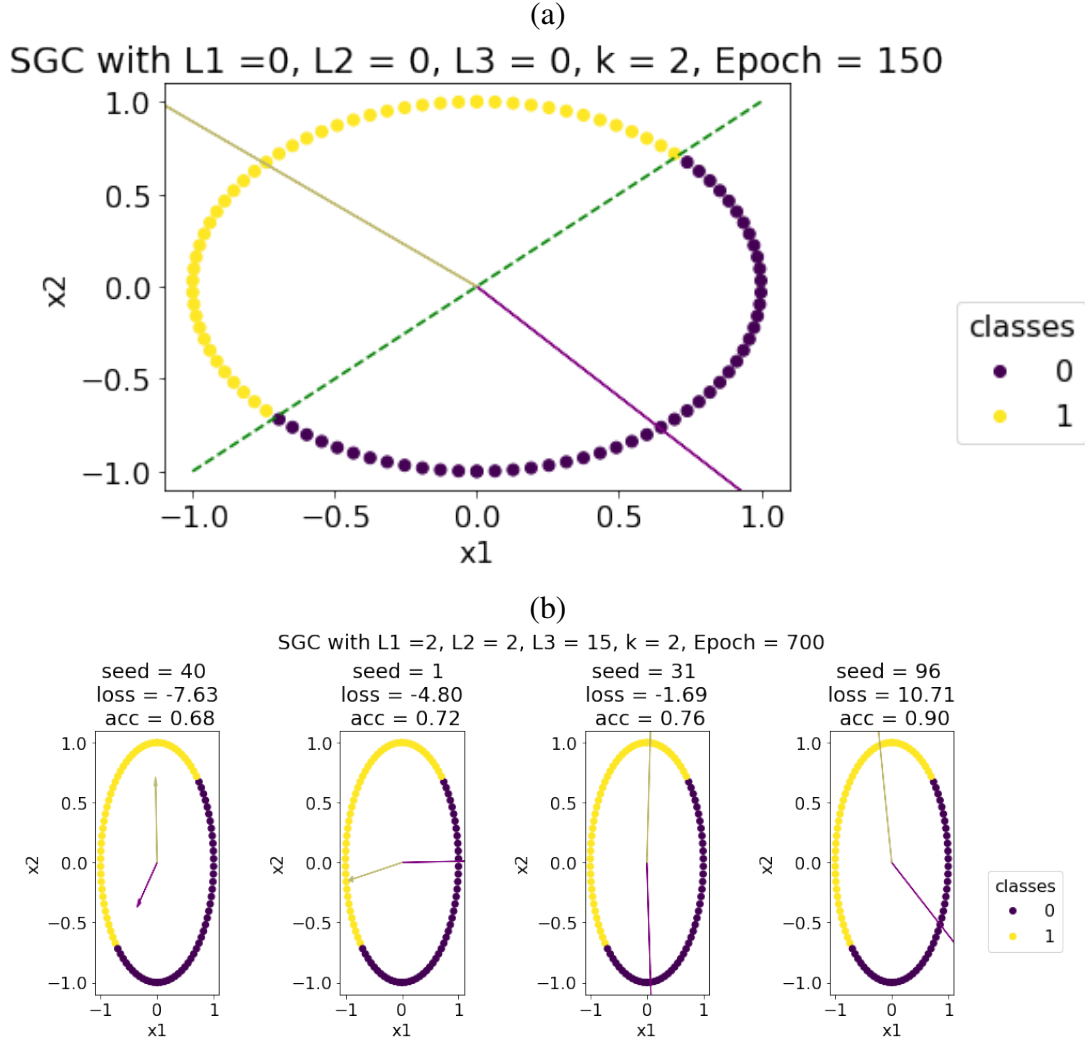


Figure 2.4: The above plots show the results of applying the SGC method *with and without regularization upon the synthetic circular data*. In Subfigure a) the dashed line shows the separation line where each side defines the point labels and the solid lines shows the SGC projections learned in Θ , and perfect accuracy is achieved. Sub-figure b) presents multiple subplots of separate independent runs of the proposed regularized SGC methodology where there are different random initializations using gradient descent. Various comparable fittings are found and it can be seen how all the aspects of the regularization upon Θ_R are respected in terms of the magnitude, relative directions between class vectors and the number of components (features) used.

accuracy (lowest loss) can be chosen.

Synthetic linearly inseparable data

In this subsection, we apply the SGC method with and without regularization on the linearly inseparable data which contains feature coordinates and a network of associations. The dataset used here is described in the Data section's subsection 'Linearly inseparable data' where the coordinate space of the datapoints and the network are displayed. The key aspect which this dataset emphasizes is that the features alone without the network information cannot produce a linear separation, but with the incorporation of the network information (with linear operators) this classification then becomes possible. Figure 2.5 shows the results of applying the SGC to the dataset without the network information being used $k = 0$, and is effectively an application of logistic regression. The methodology cannot separate the data correctly with a pair of linear projections, but this can be alleviated as seen in the following figures by incorporating the network information as well ($k > 0$).

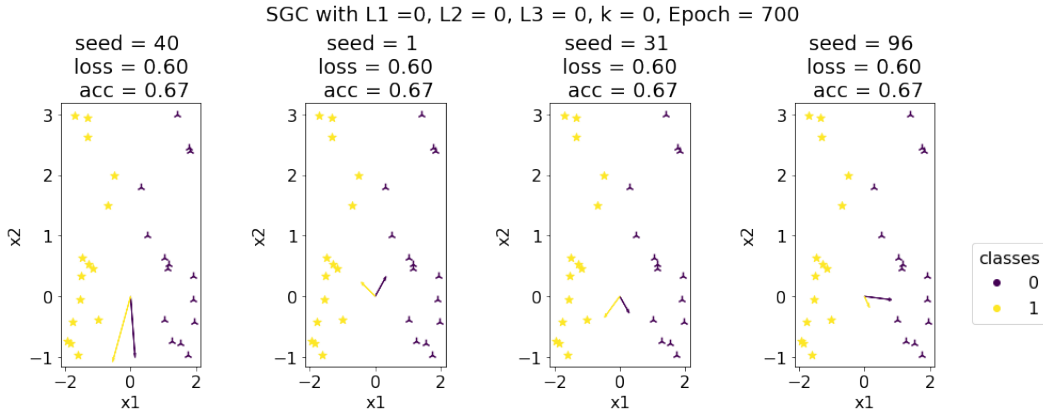


Figure 2.5: Applying logistic regression to a set of datapoints where there features are not linearly separable. This is the SGC methodology where $k = 0$ and the network information is not incorporated.

Using the SGC (by setting the shrinkage parameters to 0) Subfigure a) shows that although the vectors for the class projections, as columns in Θ , do not enable a separation between the groups

the network information enables a perfect accuracy to be produced. This is because although the support for an erroneous class can be accumulated for a point, the feature space 'communicated' to it from the edge connections of features overrides the nodes' own features in these cases. Each plot is an independent run with slight changes in Θ . Subfigure b) shows a set of plots where the axes $x1^*$ and $x2^*$ for each data point represent the projection of the features with the 'neighborhoods' of the points. With $k = 2$, \mathbf{S}^2 , aggregates the weights from '2 hops' distance in the network, so that the multiplication of $\mathbf{S}^2\mathbf{X}$ is shown on these new axes. It can then be understood why the data is then 'linearly' separable after this transformation. This emphasizes how the network information can be used to improve accuracy and maintain the simplicity of the model.

In Figure 2.7 the regularized SGC is applied to the dataset (with $L2 = 0$) and the parameter vectors for each class from Θ_R are plotted in both Subfigures a) and b). Constraints (shrinkages) are placed on the sum of the elements within $\Theta_{.,jR}$ and the direction of the vectors, which reduces the total value summation for feature extraction. In Subfigure a) the projection vectors of Θ_R are shown and as with Figure 2.6 the results produce a perfect accuracy. What can be seen is that the model explores alternative parameterizations that have not been previously found without regularization. All projections display a drop of a feature dimension. Subfigure b) shows the projection $\mathbf{S}^2\mathbf{X}$ and that perfect accuracy can still be achieved. In each of the plots, it can be seen how various equivalent (in terms of the accuracy) projections can be searched, which effectively reduce the number of features used for each class being predicted. The ability for the non-linearly separable data to be correctly classified without introduction of new parameters or 'layers' in the CNN enables explorations to be done more efficiently on large datasets in terms of time and processing capabilities.

The change of the L_3 constraint is utilized so that the projection vectors for each class are fit to be orthogonal to each other (shown in Eqn 2.9). This allows the possibility for a smaller number of classes to be fit if the class number is not known, and differs from the previous application in that

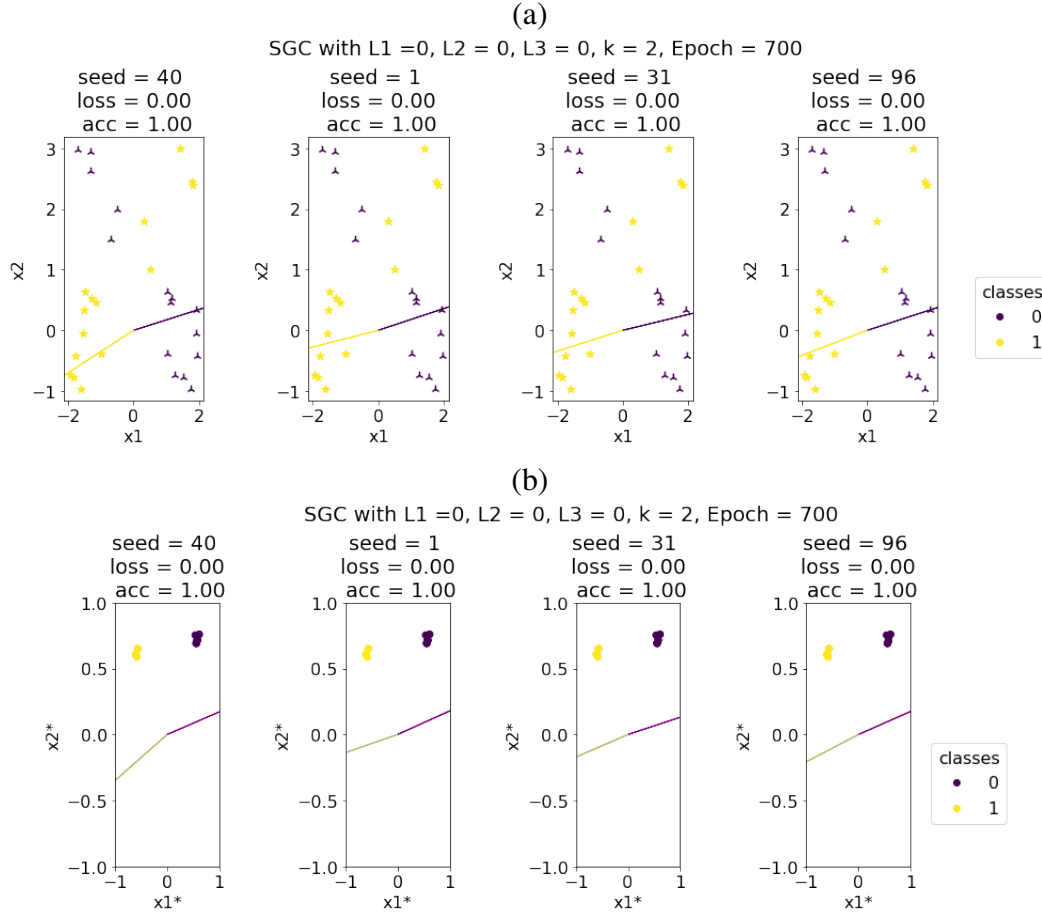


Figure 2.6: The results of applying the SGC method on the linearly inseparable data is presented here. Subfigure a) plots the data points and the 2 learned parameter vectors by SGC methodology under different initializations. It can be seen how although the displayed classification vectors within Θ with the network information provide the ability for a lossless prediction. Similarly, for Subfigure b) it can be seen how a new set of axes for the projection $\mathbf{S}^2\mathbf{X}$ (network and features) the linear separation becomes visible.

the support for each class would be seen as a separate linear function's projected value. Figure 2.8 shows the results in Subfigure a) and b) where the vector fit is seen in the space of the data points and how the data are transformed into different axes using the network data, respectively. It can be seen that the constraint for the orthogonality is preserved and the accuracy for the fits is still achieved for this problem.

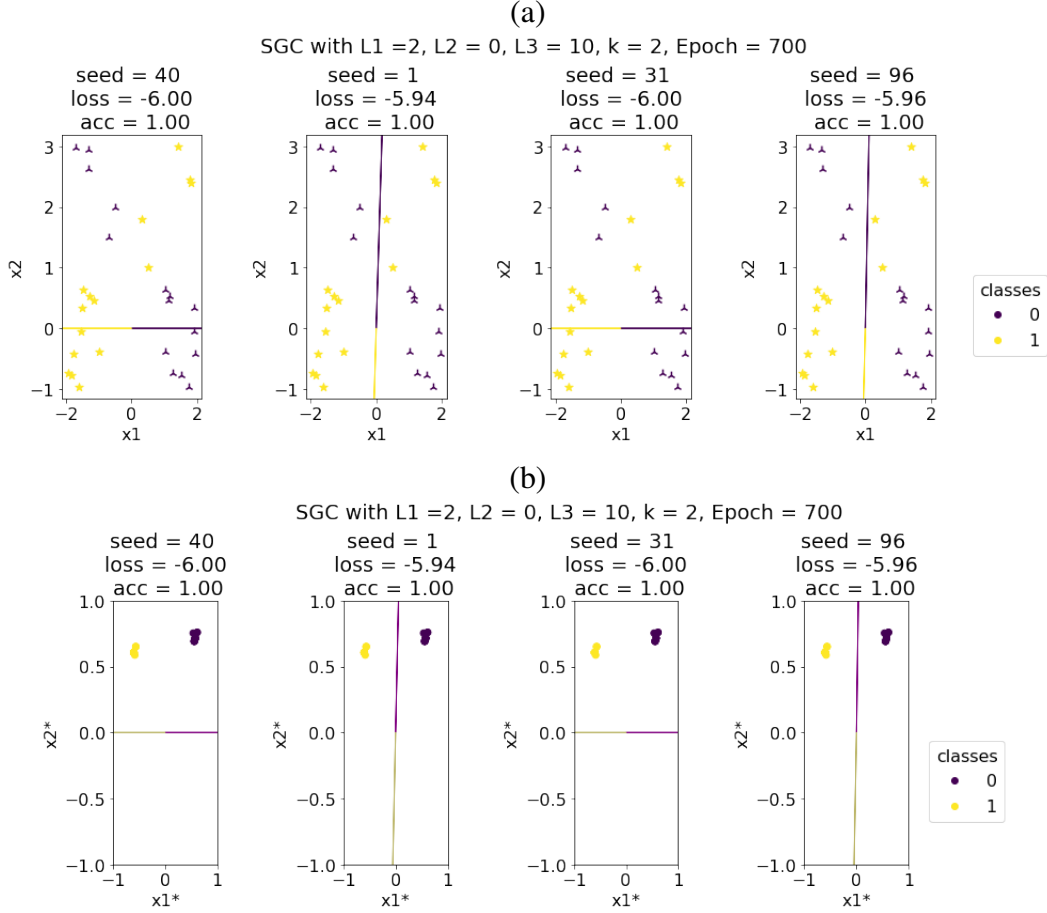


Figure 2.7: The application of the regularized SGC to the dataset where linear projections are incapable of class separation. Subfigure a) shows the parameter vectors produced by introducing constraints into SGC method on the feature space with original data points and how the classification can then produce perfect accuracy. It can be seen the proposed SGC reduces the effective number of features used in the columns of the matrix Θ_R . Subfigure b) shows the plots for the same set of weight vectors displayed on the projection axes of $\mathbf{S}^2\mathbf{X}$ where each datapoint (node) accumulates feature information from neighbors 2-hops away ($k = 2$).

Application to the Cora dataset

Here is presented the application of the SGC methodology and the proposed regularized SGC to the dataset of Cora [67]. The purpose is to examine the capability of both the SGC and the regularized SGC on a dataset with a large number of features. There are many situations in big

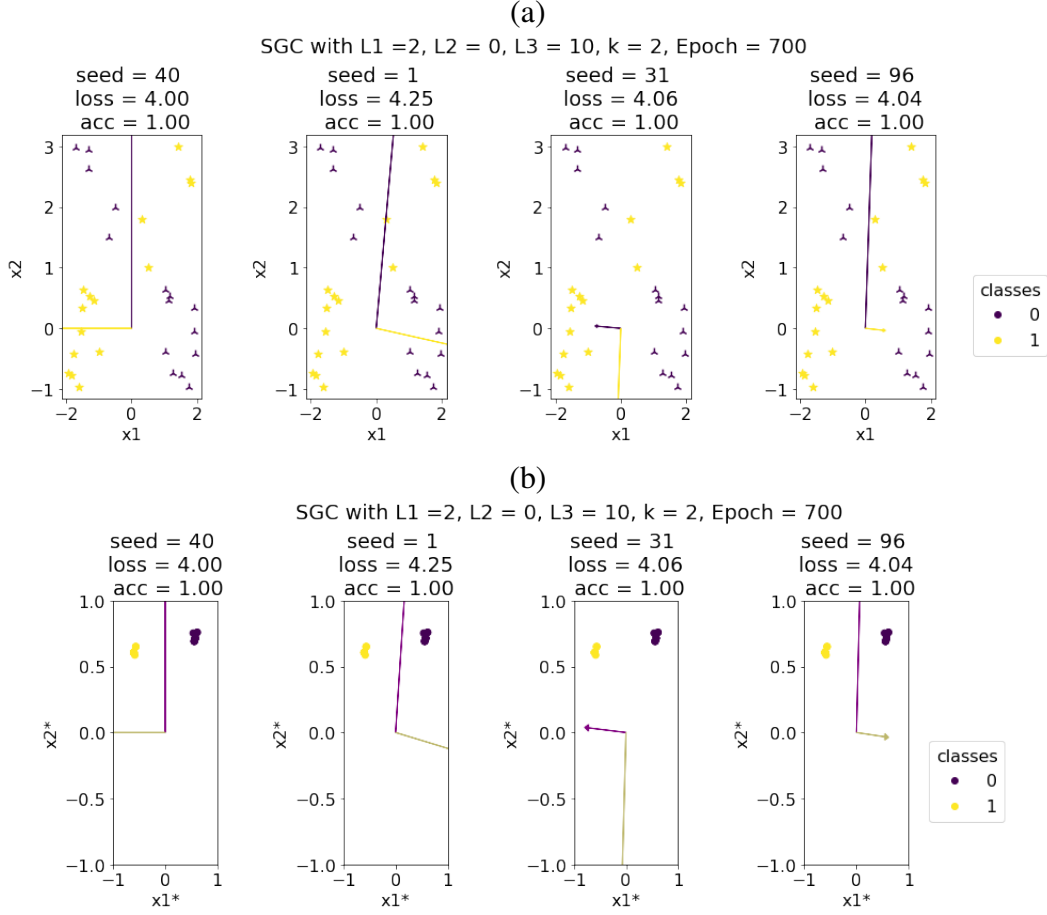


Figure 2.8: The results of applying the regularized SGC with an orthogonal constraint upon the projection achieved by using a change in regularization term L_3 . In Subfigure a) the vector projections within the methodology can be fit so that there is no loss and the orthogonality constraint is satisfied. In Subfigure b) the transformation upon the data with the network information is presented and how within this space the linear projections can separate the classes with the orthogonal vectors.

data applications where the datasets have large numbers of features due to larger data gathering schemes and a requirement to select key features without supervision. The SGC has been applied to the Cora dataset [13], and here the performance with a regularized version is mainly directed at the interpretability in highlighting the key variables in the feature set while also applying other constraints. Figure 2.9 presents the results with 2 subfigures with heat maps showing the parameter

values fitted for each class in Θ and Θ_R . The dataset classifies each document as belonging to one of seven different classes where the SGC then produces a parameter matrix Θ with seven columns and d rows for the feature number. The constraint upon L_3 is set so that the projection vectors between classes are in opposing direction so that class feature loadings are differentiated by their placement in a histogram of the values. With the SGC applied, Subfigure a) shows the weights of the parameters for each class (a single column in Θ) as a separate heatmap with a legend for the values indicated. Analogously, the same set of results produced with the regularized SGC proposed here is shown in Subfigure b). The approach produces a new parameter matrix Θ_R that introduces regularizations in the inference scheme for the parameters by penalizing their total sum and directions to be as informative about the features in terms of accuracy prediction and lack of overlap (removing redundancy within large feature spaces typical of large datasets). It can be seen that there are fewer variables highlighted for the practitioner to examine, which looks to investigate and highlight which variables are important for the class membership determination. The 7 cells highlighted in the bottom right are padding.

Using the data in the heatmaps shown in Figure 2.9 a histogram of the parameter values for each class and the features is created for the SGC and the values from the regularized SGC are held in the matrices Θ (Subfigure a) and Θ_R (Subfigure b). In Subfigure a) it can be seen how there is a smaller group of features which provide positive contribution to the class identification and that an apparent 2 mode distribution can be made out. Each plot belongs to a different class in the data set and is a different column in Θ . Subfigure b) shows the distribution of the parameter value within Θ_R . The effect of the regularization can be seen in comparison with Subfigure a) where the number of feature values at value 0 are the majority. This makes the exploration and backtracking process the features easier.

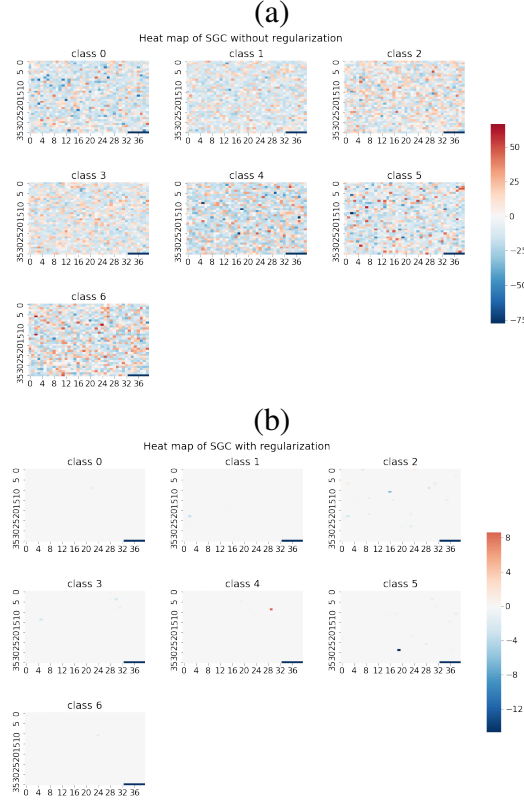


Figure 2.9: The plots showing the results of applying the proposed method of SGC with and without regularization on the features to the dataset of Cora. Subfigure a) shows the heatmap of the class columns of matrix Θ which holds the parameter values for the feature projections of the data \mathbf{X} after the inference with SGC without the regularization. Subfigure b) analogously shows the parameter values but with the inference procedure applying the constraints for the regularization as proposed which produces the shown values of Θ_R . On the bottom right of each plot there are 7 cells with padded values to produce the heatmaps. The columns of the parameter vector correspond to different classes, each shown separately, and the weights applied to each feature belonging to the nodes. It can be seen that the regularization reduces the amount of weighting over the features highlighting key variables.

Simulation Study

Simulation Setting

In this section, we investigate the performance of SGC and its regularized version under a variety of network topologies. Thirty attributed graph datasets are synthesized to imitate scale-free networks

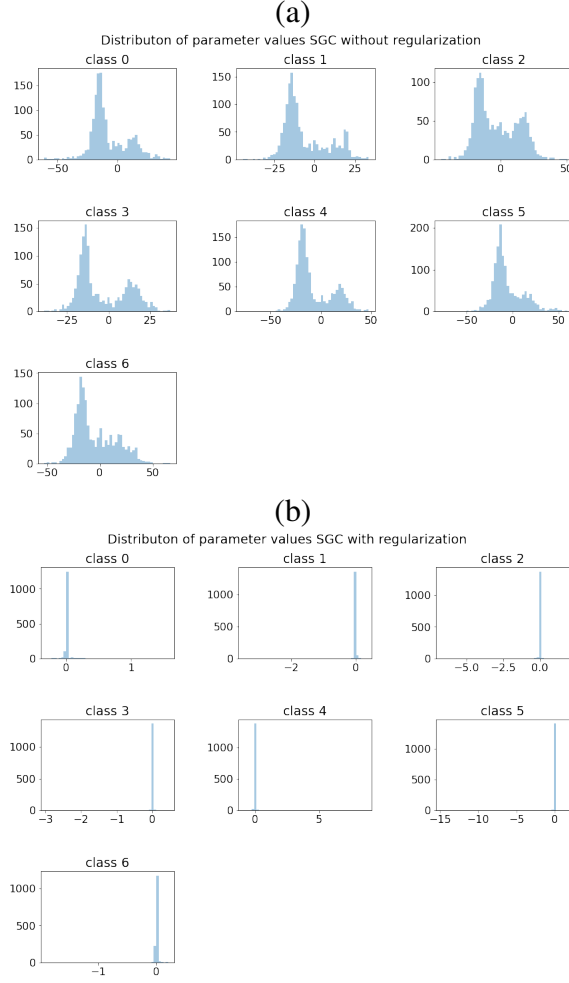


Figure 2.10: The distribution of the parameter values inferred for the Cora dataset with the application of the SGC and the regularized SGC. In Subfigure a) the SGC is applied and the histograms of the parameter values for each class in Θ is shown in the plots. Subfigure b) shows the equivalent plots but using the regularized SGC that penalizes the number of features. The majority of the features are around value zero.

which are commonly found in real applications. Each graph contains three clusters (subgraphs) with 100 nodes per cluster. Each subgraph is constructed following the Barabási-Albert preferential attachment model [11]. The interconnectivity between a pair of subgraphs is determined as follows.

- On each subgraph, a subset of nodes is chosen using weighted random sampling on degrees of the nodes. We posit that popular nodes (with high degrees) in each subgraph tends to connect with other popular nodes in other subgraph. In the context of citation network, well-known publications in one class might be cited by popular works in the other classes.
- Random edges are generated between a pair of subsets of nodes. The probability of connecting a pair of nodes is **inter_p**.

Hyperparameters (number of preferential attachment for the Barabási-Albert model, probability of random edges) are then established to control the topology of the whole graph. Samples of simulated graphs are illustrated in Figures 2.11. As the number of preferential attachment increases, the graph grows in the amount of connectivity and becomes denser. Similarly, as the inter-graph connectivity **inter_p** gets larger, more connection between clusters are generated.

The node feature matrix corresponding with each cluster is generated following multivariate normal distributions $\mathbf{X}_c \sim \mathcal{N}(\mu_c, \Sigma)$, $c = 1, 2, 3$. We set the mean vectors as $\mu_1 = (1, 0, 0)^T$, $\mu_2 = (0, 1, 0)^T$, $\mu_3 = (0, 0, 1)^T$ and the covariance matrix is $\Sigma = \mathbf{I}_3$.

For this experiment, we choose the number of hops $k = 2$ for both models. The proportion of training, validation, and test sets are 20%, 25%, 45% respectively. All models are trained for 100 epochs using the AdamW optimizer [61] with a learning rate of 0.2. We follow Algorithm 2 to tune the regularization parameters L_1 and L_3 . Optimal values of L_1 and L_3 are chosen from a grid search of 20×20 values, each varies from 10^{-5} to 10^2 , to maximize the validation accuracy. Note that we utilize the orthogonality constraint for the penalty L_3 .

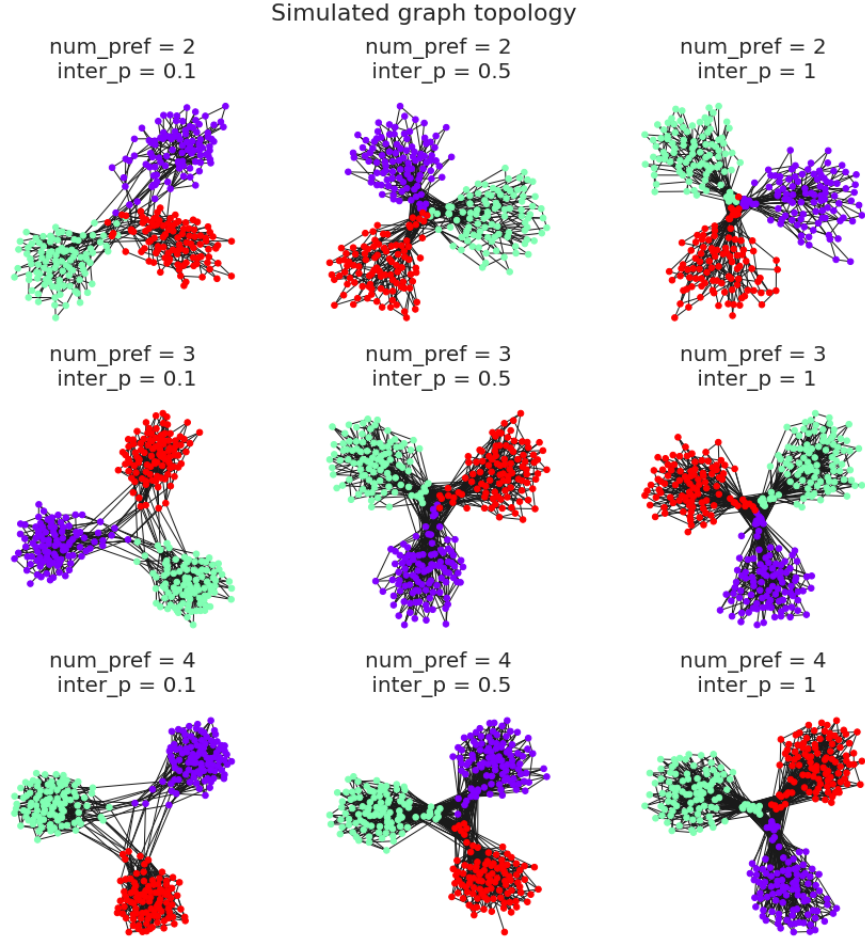


Figure 2.11: Visualizations various network topology from simulated attributed graphs, each with 3 communities (colored). Number of preferential attachment (num_pref) controls the overall amount of connectivity while the inter-graph connectivity probability (inter_p) governs the amount of linkings between subgraphs.

Simulation results

Figure 2.12 displays line graphs with test accuracy on the vertical axis and probability of intergraph connectivity on the horizontal axis. Overall, both SGC and regularized SGC achieve decent classification capability under different network structures. Our proposed framework not only performs on par with the SCG but also induces sparsity on the fitted parameters, which facilitates further

investigation on important features defining class membership.

We also observe that as the network gets denser (indicated by increasing `num_pref`), the models tend to perform better. This can be explained as on sparse graphs, nodes tend to have few neighbors, and the node representations produced by aggregating neighborhood features would become less smooth. Consequently, the models would have difficulty classifying nodes in these ineffective mapping spaces.

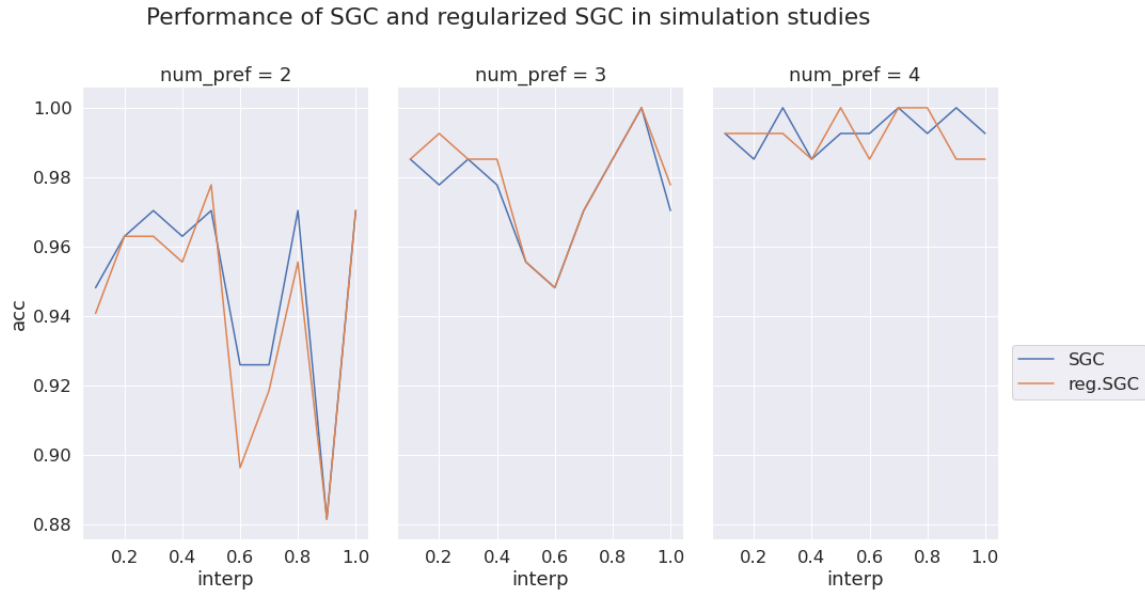


Figure 2.12: Performance of the SGC and regularized SGC under various network settings.

Discussion

We have presented a model extension of the Simple Graph Convolution (SGC) that aims at producing a smaller and more meaningful set of projections in which classification labels are presented. It addresses a key issue with interpretability of model result in big data applications, where many features may be used, which are redundant and remove the ability for a practitioner to examine

the weights. A key reason for why the SGC was chosen to be extended with this capability is that the operations are linear in the methodology, with the exception of the softmax function, so that application can be run relatively efficiently in comparison to methodologies relying on more parameterizations and more layers in order to improve accuracy.

The SGC can produce an accurate classification of data points in a feature space which is not linearly separable by utilizing the network information via linear operations. The results demonstrated this capability on a small dataset where the network projection effectively linearizes the search by having information from the node 'neighborhood' accumulated from 'k-hops' distance (relying upon the powers of the adjacency matrix). This allows for fast run times and the application to services which rely upon small delays. The methodology was applied to the Cora citation dataset, which has a large number of features, and the reduction is significant in the number of features highlighted to the user. This provides a small enough set to explore manually if required. Simulation study shows that our proposed model performs on par with the SGC on thirty attributed graph datasets with varying network structures.

Conclusion

The SGC model extension presented here allows a more explainable set of results to be presented to the user. The regularization terms reduce the number of non-zero parameters and the overlap between parameterizations of the different classes. Future work could entail a more in-depth exploration of how the network can be 'decomposed' in such a way as to minimize the number of label alterations. Producing a network separation by eliminating edges can find applications in social networks where polarized communities must be isolated as a means of inoculation.

CHAPTER 3: EXPLORING THE VALUE OF NODES WITH MULTICOMMUNITY MEMBERSHIP FOR CLASSIFICATION WITH GRAPH CONVOLUTIONAL NEURAL NETWORKS

Introduction

The rapid growth of graph data with millions of nodes makes inspecting these elements individually for generalization or inferring underlying relationships a challenging task. Simplifying the vast collection of nodes into a handful of communities or groups facilitates the investigation of network structure. This simplification process can be regarded as assigning nodes with distinct labels of their communities capturing aggregated behaviors of all existing nodes in the network. Examples for the label application can cover voting patterns which are pigeon holed into a small number of choices, consumer buying patterns in respect to certain products and even different psychological profiles. This concept is applied in algorithms using collaborative filtering [92] (usually for retail) where recommendation systems apply a customer's interests to find the closest community set to predict an affinity for new items. The principle underlying the ability to group nodes together in this fashion is based on a degree of homophily [68] in the groups. Examples of this are found in the work of [44], which studies how social network connections created from friendships or interests can drive political engagements differently. For the growth of a network where edges are constructed, nodes create connections or affiliations and it becomes a question of determining the label for a node with which an edge is constructed. Choosing these connections becomes an important issue for the originating node as it produces label associations.

Traditional statistical approaches, such as logistic regression, use only node features to infer their labels and ignore node connectivity information. On the other hand, network science tools such as

community detection algorithms (e.g. Louvain algorithm [15]) only make use of the placement of a node in the network topology, but they do not take into account the node features for allocating the labels. The methods of Graph Neural Networks (GNNs) [123] provide a framework which combines both the feature information and the network information in order to make inferences on the labels applied to nodes.

The methodology of the Simple Graph Convolutional (SGC) [121] (described in more detail in Methodology section) presents an intuitive, simple, and expressive formulation for learning these latent representations of the nodes labels which builds upon the general theory of graph convolutional networks [131]. This methodology is appealing because the operations are linear between the adjacency matrix, the features, and the parameters prior to the use of the softmax function. This makes it an ideal candidate to work with in exploring different applications of its formulation, as the feature projections are linear and the adjacency matrix is clearly an operation aggregating feature information of the vicinity of the nodes.

A machine learning pipeline usually consists multiple parts: sample, explore, modify, model and assess [8]. As the data travel through the pipeline, the error propagates; a mistake in an earlier step may have resounding impacts on the pipeline performance. Active learning focuses on the sampling step by prioritizing samples that are assumed to be more helpful to the task [95]. Sampling the training corpus prior to training the model is conducted to attempt to receive similar, if not better, accuracy utilizing fewer data. This process has been shown to be successful in multiple domains, such as natural language processing and image data [31, 102, 97, 107]. Recent work has been conducted on the performance of active learning on graph data [62, 122]. In general deep learning has shown great advantages in many fields, [133, 51, 134], and spatial components for a complex object [113].

For graph data, there are a variety of node-ranking algorithms such as PageRank [75] and VoteRank

[128]) which can be utilized to select nodes in the active learning process. In this paper, we look for optimal sampling methods for the node classification task across five real graph datasets which span a large domain of network topologies. Our main contribution is the discovery of the effect of the sampling method or direction (i.e., ascending versus descending selection) on the results of a node classification task.

This paper looks at how the SGC can be used to arrive at correct label allocations of nodes in networks (where nodes contain features) where the full set of the network nodes is not provided. The context is that often the full network is not visible to the investigator and that the set of nodes are actually sampled. From the investigation, it can be seen that how nodes are sampled can change the accuracy of labels trying to be predicted. It could be assumed that nodes which have the largest number of edges would be the best representatives of a community label so that their label would propagate to other nodes for whom it is the most central node. The investigation here shows in the results that counterintuitively low ranked nodes in terms of centrality can provide better information on the labels. The fact that the lower-ranked central nodes contain more accurate information for label allocation supports the general idea that *weak-ties* are valuable [18]. Two strategies are proposed to predict the best sampling methods based on the network topology. First, we find that the skewness of homogeneous connectivity distribution is an accurate predictor for the sampling direction. Furthermore, we empirically find a correlation between the topology structure, consolidated by a single statistic, and the sampling direction. Our contributions are listed as follows:

- We introduce sampling strategies to improve the performance of node classification using SGC framework
- We propose indicative measures built upon network topology facilitating users in choosing sampling methods for their applications

Data

An attributed graph $G = (X, A, y)$ is represented by three components: an adjacency matrix $A \in \mathbb{R}^{N \times N}$, a feature matrix $X \in \mathbb{R}^{N \times D}$, and a node label vector $y \in \mathbb{R}^N$. Real datasets were gathered from online resources; seven of the nine datasets were accessed using open-source python libraries [98], [118]. The other two, Lastfm-Asia and Deezer-Europe, were downloaded from the Stanford Network Analysis Project's repository [56].

Cora [67], Citeseer [33], and Pubmed [94] are three citation datasets where scientific publications are nodes, and the references between them form edges. Each paper is represented by a binary vector indicating the presence of words in the paper (features) and categorized by the corresponding topic (labels).

Coauthor-CS [98] and Coauthor-Physics [98] contain co-authorship information in computer science and physics academic publications respectively. The authors are considered as nodes, defined by a vector of keywords in their published papers. Their coauthorship forms edges. Each author is categorized by their most active field of study.

Amazon-PC and Amazon-Photo are subsets of the Amazon co-purchase graph [66] where goods are considered as nodes represented by a vector of keywords in their product review and classified by their product category. Connection between a pair of nodes (the edge) indicates that these products are frequently purchased together by customers.

Lastfm-Asia and Deezer-Europe are social network datasets introduced in [88]. In Lastfm-Asia, nodes are social network users of LastFM, defined by their artists-of-interest, and edges are formed by their mutual followers. Each user is classified by location. Deezer-Europe is a network of Deezer users from European countries (nodes) and their mutual followers (edges). Each node is represented by their artists-of-interest (features) and categorized by the user's gender (labels).

Dataset	Nodes/Edges/Classes	Description
Cora	2708/5278/7	Scientific publications.
Citeseer	3327/4614/6	Scientific publications.
Pubmed	19717/44325/3	Diabetes-focused scientific publications.
Amazon-PC	13752/287209/10	Computer goods sold at Amazon.
Amazon-Photo	7650/143663/8	Photos sold at Amazon.
Coauthor-CS	163788/18333/15	Authors of computer science papers.
Coauthor-Physics	34493/495924/5	Authors of physics papers.
Lastfm-Asia	7624/27806/18	LastFM social network users.
Deezer-Europe	28281/92752/2	Deezer European social media users.

Table 3.1: Dataset statistics and domain-specific information.

Methodology

Sampling methods

Two procedures of sampling are considered in this study, namely descending and ascending. In descending sampling, training instances are selected by gradually acquiring from the most important nodes to the least important ones. On the contrary, ascending sampling gradually selects training samples starting from the least important nodes to the most important ones.

Three different criteria are used to evaluate a node’s importance (centrality) for sampling orders.

Degree

In degree sampling, we acquire nodes for training based on their corresponding number of directly connected neighbours (i.e. node’s degree).

PageRank

PageRank algorithm [75] derives a web page (node)’s rank by accumulating its incoming neighbors’ ranks proportionally to their total number of outgoing connections. The resulting ranking represents the relative importance of pages in the network. In this study, we apply PageRank to rank all the nodes in our graphs and then sample them based on their rankings.

VoteRank

VoteRank algorithm [128] iteratively selects a set of important nodes called spreaders using voting scores given by the neighboring nodes. Once a node is selected as spreader, it is excluded from next round of voting and its direct neighbors’ voting abilities are also reduced. In this study, we employ VoteRank to all nodes in the graph (by setting the number of spreaders as the total number of nodes) and then sample them based on their rankings.

Simple Graph Convolution (SGC)

SGC [121] is a simplified GNN model developed from GCN [48] by removing non-linear activation functions between hidden layers and reparametrizing successive layers into one single layer. This simplification reduces superfluous complexity of GCN while retains superb performance on many downstream tasks. The work of [79] illustrates SGC’s expressive power in node classification task and proposes a flexible regularization methodology to reduce the number of parameters and highlight a sparse set of important features. The SGC is a ‘one-shot’ learner which simplifies the training procedure and allows for the full set of data points to be used for the parameter inference.

In this section, we briefly present the original SGC. An attributed graph data set contains a graph $G = (V; \mathbf{A})$ and a feature matrix $X \in \mathbb{R}^{N \times D}$. The graph G composes of $V = (v_1, v_2, \dots, v_N)$ is a set of N nodes (vertices) and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix where each element a_{ij} represents an edge between node v_i and v_j ($a_{ij} = 0$ if v_i and v_j are disconnected). We define the degree matrix $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_N)$ as a diagonal matrix whose off-diagonal elements are zero and each diagonal element d_i capture the degree of node v_i and $d_i = \sum_j a_{ij}$. Each row x_i of the feature matrix $X \in \mathbb{R}^{N \times D}$ is the feature vector measured on each node of the graph. Each node i receives a label from C classes and hence can be coded as one hot vector $y_i \in \{0, 1\}^C$.

The GCNs and SGC add self-loops and normalize the adjacency matrix to get the matrix \mathbf{S} :

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \quad (3.1)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \text{diag}(\tilde{\mathbf{A}})$. This normalization allows successive powers of the matrix to not influence the overall size the projections. The SGC removes nonlinear transformations from the k^{th} layer of the GCN, resulting in a linear model of the form:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S} \dots \mathbf{S} \mathbf{X} \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}). \quad (3.2)$$

The SGC classifier is then achieved by collapsing the repetitive multiplication of matrix \mathbf{S} into the k^{th} power matrix \mathbf{S}^K and reparameterizing the successive weight matrices as $\Theta = \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}$, and its structure as a GNN is defined by:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S}^K \mathbf{X} \Theta). \quad (3.3)$$

The parameter k corresponds to the number of 'hops' which is the number of edge traversals in the network adjacency matrix \mathbf{S} . k can be thought of as accumulating information from a certain

number of hops away from a node (as visually described in [121]). If $k = 0$ the methodology becomes equivalent to a logistic regression application which is known to be scalable to large datasets. Since the SGC introduces the matrix \mathbf{S} as linear operation, the same scalability applies. The weight matrix Θ is trained by minimizing the loss:

$$\mathcal{L} = \sum_{l \in \mathcal{Y}_L} \sum_{c \in C} Y_{lc} \ln \hat{Y}_{lc} \quad (3.4)$$

where \mathcal{Y}_L is a collection of labeled nodes. This model allows for a very computationally efficient exploration of the network based datasets but this multilayer approximation may not provide the full extent of deep learning generalizations.

Evaluation of Network Topology

The network topology was evaluated using the inverse coefficient of variation of the node's degree distribution.

$$iCV_d = \frac{\mu_d}{\sigma_d} \quad (3.5)$$

where $\mu_d = \frac{1}{N} \sum_{i=1}^N d_i$ is the average degree and $\sigma_d = \frac{1}{N-1} \sum_{i=1}^N (d_i - \mu)^2$ is the standard deviation of degree.

A low value of iCV_d occurs for networks that have a high variation in their degree distributions compared to the mean degree. It indicates that important hubs (nodes) are highly connected to other nodes. On the contrary, a high value of iCV_d results from relatively low variation in degree distribution compared to the mean degree where important nodes tends to be less popular.

The node degree centrality is defined by

$$D_i = \frac{d_i}{\max(d_i)} \quad (3.6)$$

where d_i is the degree of node i .

Homogeneous connectivity is the proportion of homogeneous connections that a node has normalized by its total number of connections.

$$\Omega_i = \frac{h_i}{d_i} \quad (3.7)$$

where h_i is the number of homogeneous nodes within one degree.

Results

Sampling the nodes from a graph causes a change in performance. Active learning [95], as opposed to passive learning, selects a set of most informative instances for training to achieve the best performance while using minimal samples. This paper utilizes common sampling methods (i.e. degree, PageRank, VoteRank) to select nodes in a graph for the training corpus using an ascending (i.e. lowest to highest score) and descending (highest to lowest score) fashion.

Two separate methods are formulated to predict the best sampling direction. A method that utilizes the network topology looks for a partition in the iCV_d domain. The pros of this method is that it utilizes information which is readily available prior to classification, excluding the requirement of knowing the ground truths. A second method is proposed which compares the skewness of the homogeneous connectivity distributions to evaluate the best sampling direction. This method utilizes the ground truth to explain why the best sampling direction is, in fact, the best.

Two different types of plot are visualized for each dataset. The sampling result plot shows the performance of the node classification task (measured by accuracy) on various training sizes across multiple sampling techniques. The accuracy curves tend to improve as more training samples are recruited. The box-plots show results of random sampling with 10 replications. With further inspection, sampling methods (plotted as lines) can be viewed to pass specific judgement regarding its performance. If these sampling methods show better performance than random selection, it can be concluded that the method is an improvement.

Degree centrality plot composes of a scatter plot illustrating the relationship between homogeneous connectivity, Ω_i (Equation 3.7) node centrality (Equation 3.6) (on the left panel), and a histogram showing the distribution of homogeneous connectivity (on the right panel). The purple points describe the nodes that are sampled following the "degree ascending" method when $s = 0.5$; similarly, the yellow points describe the nodes that are sampled following the "degree descending" method when $s = 0.5$. Because the scatterplot is visualized according to node degree centrality on the x-axis, a clear partition is found between the purple and yellow points. It is also informative to look at the histogram, which shows the distribution of the homogeneous connectivity according to the sampling direction. On this plot, some deductions can be made as to why the superior sampling direction was more effective.

In a high iCV_d graph (Cora and Citeseer), the three ascending methods almost uniformly render higher accuracies than the ascending methods across training sizes (Figure 3.1). The dominance of descending sampling in these graphs could be explained by the fact that important (central) papers of certain disciplines are usually cited by many papers in the same discipline. Consequently, the most important nodes contains crucial information about the class label and hence is beneficial for node classification task.

The distribution of homogeneous connectivity across the Cora and Citeseer data sets is similar

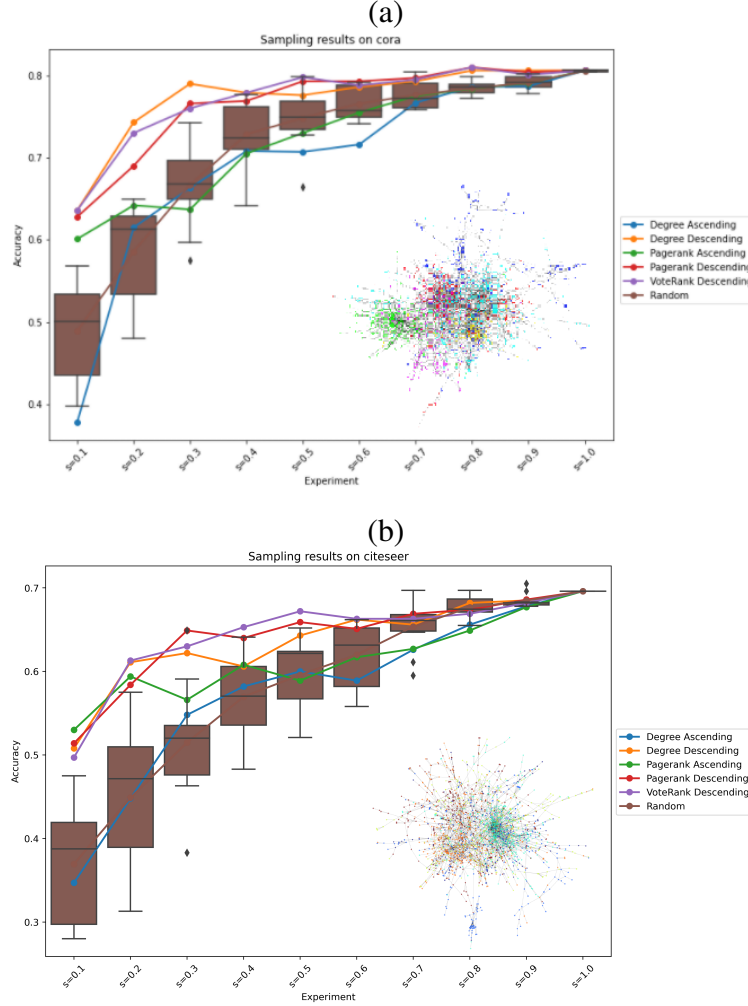


Figure 3.1: High iCV_d networks (Cora, Citeseer) have higher accuracies when sampling nodes from the highest score to lowest score (i.e. 'descending' methods), showing the effectiveness of the node ranking algorithms on a node classification task.

(Figure 3.2). Homogeneous connections of both data sets exhibit left-skewed patterns indicating the existence of clusters of informative nodes (high Ω) and noisy nodes (low Ω). Nodes with low Ω mainly connects with neighbors across different categories while node with high Ω mostly connects to neighboring nodes within the same category.

On Cora, the ascending and descending samplings possess a similar amount of most informative

nodes. However, the bottom 50% of the central nodes shows a higher left skewness (3.2A)). It indicates that less popular papers are noisier, since they tend to get cited by papers in different categories. Hence, recruiting these samples in the training step is not desirable, since they provide noisy representations of the corresponding categories and deteriorate the performance of the classifier.

On Citeseer, a different pattern occurs where large amounts of noisiest nodes exist in both sampling schemes. However, the descending samplings contain a higher amount of moderate to high informative nodes, as the distribution of the top central nodes exhibits lower degree of left skewness (3.2B)). Hence, descending sampling tends to work better since recruiting popular papers provide smoother representation of their categories

Alternatively, we observe an opposite trend in low iCV_d graphs (Pubmed), where ascending samplings prevail (Figure 3.3. Pubmed citation graph contains publications on diabetes and, hence, has a smaller scope compared with other citation data sets. Important (central) papers might be cited by other papers across classes due to the close nature of their categories. Therefore, important nodes contains a less differentiating factor for classification tasks. On the other hand, less important nodes might contain unique characteristics of the class and render useful information for node classification task.

Pubmed's homogeneous connectivity distributions are highly left-skewed (Figure 3.4). Both sampling schemes contain relatively high amounts of informative and noisy nodes. Descending sampling has relatively higher skewness implying a heterogeneous selection of high quality and low quality popular papers (in terms of their homogeneous connectivity). Popular papers (high D) with low amount of within category citations (low Ω) get cited by other papers from different category. Hence, the descending strategy has a worse performance since these low-quality popular papers inevitably induce a confusing representation of the category.

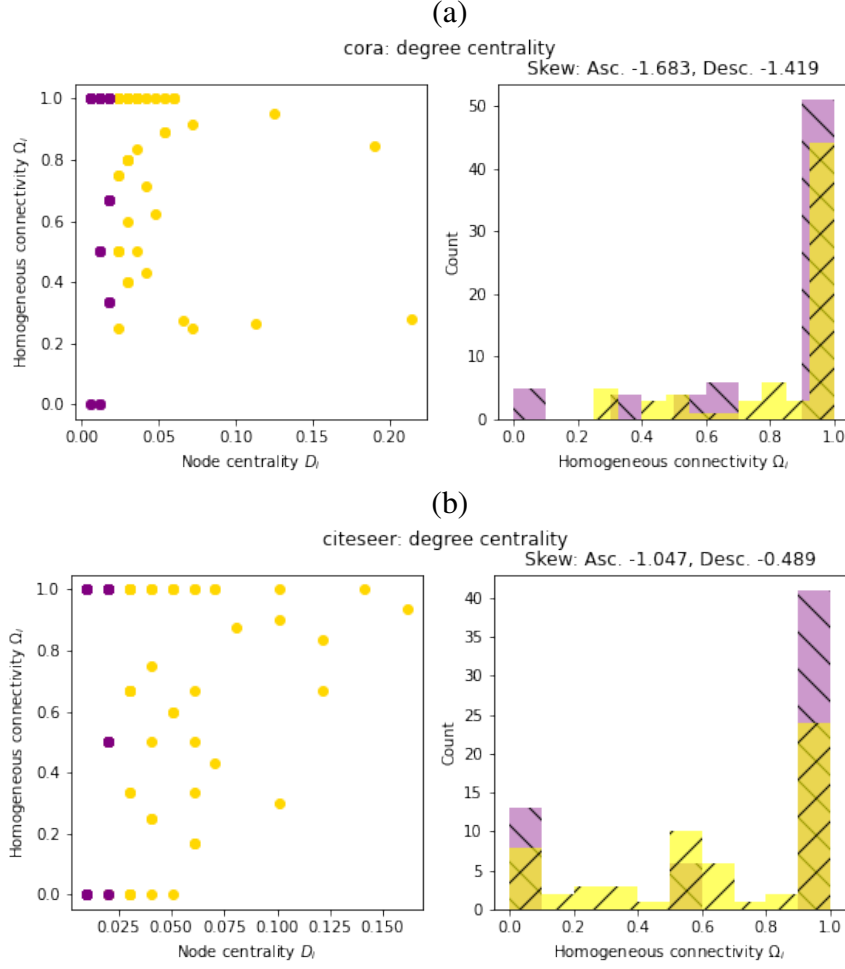


Figure 3.2: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skewness for each subset's distribution is annotated above the right graph.

The LastFM-Asia is a social media data set which categorizes users based on their country of origin (Table 3.1). Node classification results change dramatically with training size, s . After $s = 0.3$, the ascending sampling methods perform consistently better than the descending methods. On this social network, nodes with smaller importance are more indicative of a node's label, the person's country of origin. Users with smaller followships could be more likely to be connected with

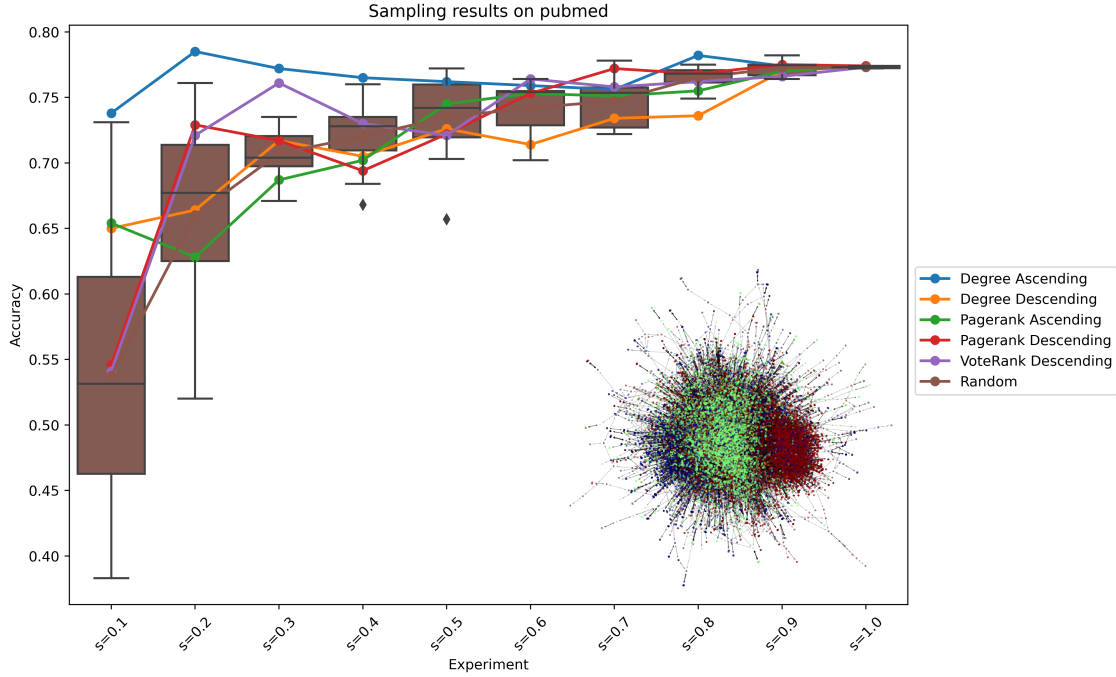


Figure 3.3: Low iCV_d networks (Pubmed) have lower accuracy when sampling nodes from the lowest to highest score (i.e. 'ascending' methods), showing the ranking algorithms are inversely beneficial to the node classification task.

people they know personally, within their real-life circle. However, people with more followers are more famous and likely have more followers across the globe, therefore causing the country to be hard to discern. Much like the other data sets, the homogeneous connectivity distribution for LastFM-Asia is left-skewed. In other words, there exists a large set of edges that are interconnected within the community and fewer that are connected to other communities. The skewness of the ascending selections is greater than that of the descending selections; as a result, the utilization of ascending rank, as the sampling direction, is chosen, which matches with the node classification results.

The Deezer-Europe social media data set shows varied results. Most of the sampling methods, agnostic to the sampling direction, consistently perform better than random. In other words, the

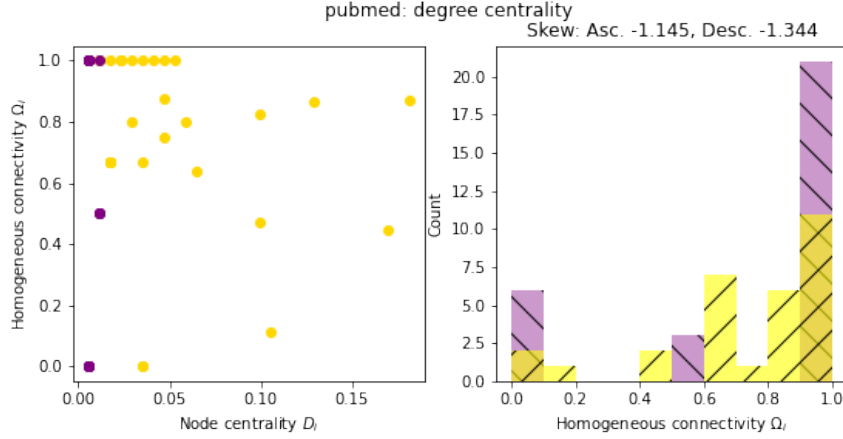


Figure 3.4: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skew for each subset’s distribution is annotated above the right graph.

more popular and less popular nodes are helpful in the node gender-classification task, as opposed to users in the middle-ground. Intuitively, the networks of less popular users likely respond to gender homophily, as shown in certain age groups in [49]. More popular users likely have growing followerships which can be based on mutual interests, especially in this network’s musical context. The Deezer-Europe dataset renders a unique homogeneous connectivity distribution, showing one which appears to be Gaussian. The average value is around 0.50. The skew of the Ω_i domain is in favor of the descending sampling process, which is also the conclusion made by the iCV_d process.

In this paper, we have found a correlation between network topology and the optimal sampling strategy. Interested readers are referred to the appendix for the results of the remaining datasets that are not discussed here. This fact implies that machine learning practitioners can deduce an optimal sampling strategy by 1) evaluating their network topology, and 2) observing its position in Figure 3.7. The results show that no sampling method is superior in terms of accuracy; the logistic probability of a descending sampling evaluation producing the best results increases with increas-

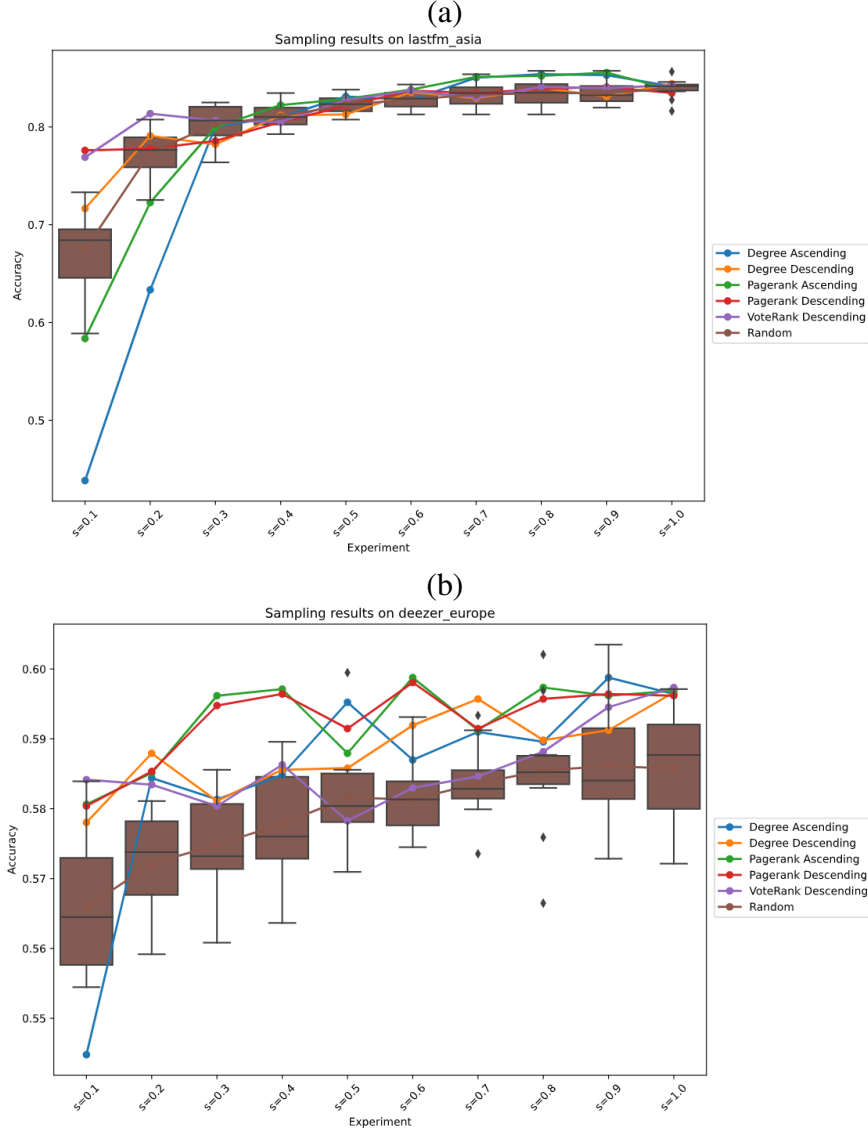


Figure 3.5: The performance of this pipeline on the Deezer_Europe social media dataset (plot b) is unusual in that almost all sampling methods are uniformly better than random selection.

ing iCV_d . A high iCV_d occurs for highly connected graphs (high μ_d) where all nodes have a similar number of connections (low σ_d). Coauthor-cs, citeseer, and deezer_europe are among the highest scorers in iCV_d . In these graphs, important nodes have relatively lower popularity, which correlates with a descending sampling direction because these nodes contain the defining characteristics of

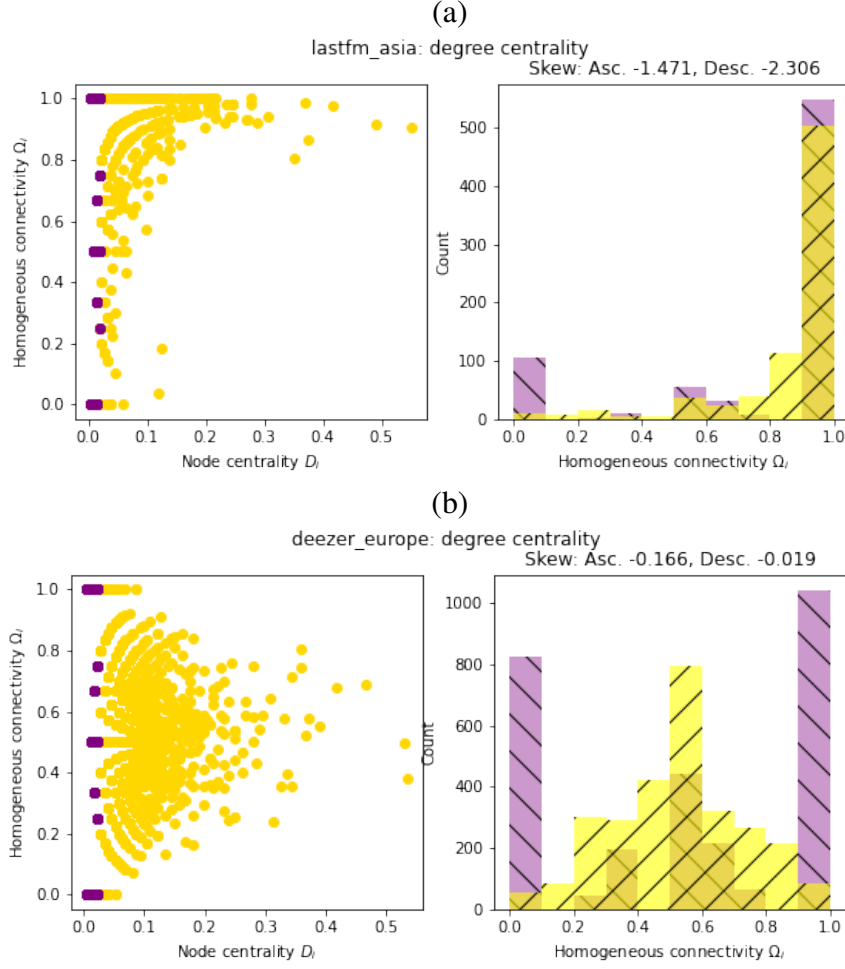


Figure 3.6: The left figure here presents scatter plots of node degree centrality, D_i against node homogeneous connectivity Ω_i on the training data. The upper half of nodes according to their centrality are colored in yellow while the lower half is presented in purple. The histogram on the right visualizes the distribution of homogeneous connections. The skew for each subset's distribution is annotated above the right graph.

their associated categories. A low iCV_d occurs for low connected graphs where important nodes are highly connected, i.e. more popular which correlates with an ascending direction. Sampling less popular nodes is more beneficial, since they contain distinct characteristics to represent associated categories.

Network topology-informed sampling methods (i.e. all methods except random) seem to perform

well in node classification tasks, often resulting in similar accuracies utilizing a smaller amount of data. Additionally, independent of the ascending/descending, we see across the board a higher number of cases where the more complicated sampling procedures (i.e. Pagerank/Voterank) outperform Degree. While we see an increase in performance, there is a trade-off with computation time; nodes degree distribution can be computed swiftly while PageRank and VoteRank require complex evaluation, and hence be more computationally expensive.

Dataset	Prediction	Actual
Cora	Descending	Descending
Citeseer	Descending	Descending
Pubmed	Ascending	Ascending
Amazon-pc	Descending	Ascending
Amazon-photo	Ascending	Ascending
Coauthor-cs	Descending	Descending
Coauthor-physics	Ascending	Descending
Lastfm_Asia	Ascending	Ascending
Deezer_Europe	Descending	Descending

Table 3.2: The sampling direction is predicted with a high accuracy by studying the skewness of the homogeneity connectivity distribution. Misclassifications are likely caused by a lack of node importance evaluators that are robust to graph topology.

The skewness of the homogeneous connectivity Ω_i distribution is indicative of the better performing sampling direction. Left-skew distribution is more common because it is expected that there is an association between network topology and nodes’ label, in which nodes with the same labels tend to connect with each other. Graph neural networks utilize message passing to learn expressive node embedding for a given task [34]. The mechanism involves aggregating features of a node’s neighbors to produce a smoother representation where neighboring nodes tends to have similar property such as belonging to the same class. Therefore, a left skew of Ω_i is suitable for graph neural networks to learn the effective node embedding for a classification task.

Under the assumption that a Ω_i is left skewed, the sampling method which renders a weaker skew-

ness will be the one which performs better. A stronger left-skewed distribution has an elongated tail, which recruits more noisy, low informative nodes. These samples aggregate features of neighboring nodes belonging to other classes and provide a poor representation of their own classes. Their noisy representations inevitably induce more confusion to the model and degrade the performance on classification task. Table 3.2 demonstrates the agreement between skewness of the homogeneous distribution and the best sampling approaches.

Some graphs (i.e. amazon-pc, coauthor-physics) do not robustly fit the node importance evaluators utilized in this study, as indicated by the poor performance of informed samplers compared to random sampling. Both examples show conflicting results when using our two sampling direction detection schemas; coauthor-physics concludes ascending via Ω_i and descending via iCV_d and amazon-pc concludes descending via Ω_i and ascending via iCV_d . Future work will be required to observe the domain in which this phenomenon occurs, since both examples of conflicting indications occur at the edges of the iCV_d domain. Given that node degree is one of the measures of node centrality, we would assume that using other centrality measurement (such as VoteRank) might render harmonious conclusions of sampling schemes from Ω_i and iCV_d .

In practice, obtaining homogeneous connectivity distribution prior to sampling is impractical since it requires knowledge about the labels in the calculation process. Hence, we developed an alternative criteria to help the practitioner select the best sampling approach based on the inverse of the coefficient of variation of the node degree. Our experiments show an relationship between the network topology (summarized by iCV_d) and the best sampling direction (Figure 3.7).

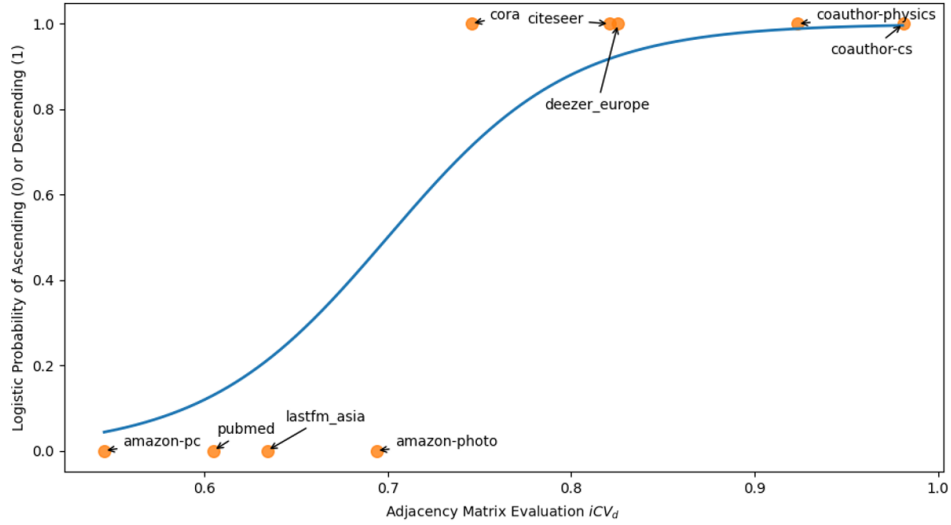


Figure 3.7: Logistic probability (blue line) shows an increasing likelihood of a descending sampling procedure as the coefficient of variance of the degree (iCV_d) increases. Results show a complete separation is defined by iCV_d .

Discussion

The careful selection of nodes for a machine learning process helps increase the accuracy of correct label prediction, enticing the study of different sampling methods. Two sampling methods are evaluated, ascending and descending, in which samples are selected based on node centrality as defined by three different measures (Voterank, PageRank, and degree). We conclude that there does not exist a uniformly best method for node selection across all network topologies.

Before building a classification pipeline, it is useful for the practitioner to have an estimate of which sampling direction is superior. An indicative measure of best sampling strategy is the skewness of homogeneous connectivity distribution. A left-skewed distribution is desirable since neighboring nodes tend to belong to the same class and hence produce smoother representation for the node classification task. However, we found that a strong left skewness, indicating a selection of more

noisy and low informative nodes, is detrimental to the performance of the classification task. However, rendering the homogeneous connectivity is impractical for practitioners due to its reliance on knowing the node's labels. Therefore, we present a second method that only requires network topology information (iCV_d). This method is empirically proven.

Future work will apply these findings to large social media networks for tasks like job searching. Further, applications to knowledge embedding in the natural language processing domain will be pursued.

CHAPTER 4: EXPLORING A LINK BETWEEN NETWORK TOPOLOGY AND ACTIVE LEARNING

Introduction

In the previous chapter, we have discovered that network topology has a significant impact on the effect of sampling strategy for the node classification task. A heuristic criteria based on node degree, the inverse of the coefficient of variation of node's degree (iCV_d), has been introduced as an indicative measure for practitioners to select the best sampling strategy. In this chapter and in [38], we further evaluate the effect of iCV_d and derive effective sampling approaches to facilitate the process of predicting node label.

Labels of nodes can be inferred using standard classification methods, such as logistic regression, which are predominantly reliant on the node feature information after a training phase. However, these methods do not consider the supplementary node connectivity information. Additionally, community detection algorithms (e.g. Louvain [15]) take into account node connectivity information, but not node feature information. Graph neural networks (GNN) combine both information into a framework for inference (i.e. label prediction). The Simple Graph Convolution (SGC) [121] simplifies GNN to a logistic-regression-like formulation while maintaining the connectivity information of the node. The computational efficiency of this model allows for the practical experimentation done in this study.

In many cases, labeled data is limited and costly to produce. The field of active learning focuses on ordering the available labeled data prior to the training process for the purpose of strategically showing the model more informative nodes earlier, allowing it to generalize with less data while maintaining a similar (or superior) performance [96]. This project focuses on the application of

active learning to graph neural networks (GNN) by utilizing available node ranking algorithms such as node connectivity densities (i.e. degree), PageRank [75], and VoteRank [128]. Similar to [39], this work experiments with bidirectional sampling (i.e. ascending and descending) of these algorithms' rankings.

Node classification task across four real graph datasets are optimized using the six node selection processes (i.e. ascending & descending selection along 3 node importance evaluators) to study the correlation between the superior sampling process and network topology. Results show that the sampling direction (i.e. ascending vs descending selection of samples with respect to their importance rankings) is dependent on network topology. The results are empirically reverse engineered using an unsupervised process to allow the prediction future applications to derive the best sampling method as opposed to the brute force experimentation provided in this study. Generally, networks with sparse topologies perform better in node classification tasks when the active learning process uses a descending node selection; conversely, dense networks prefer ascending node selection. Our contributions are as follows:

- We design comprehensive synthetic attributed graph data to examine the heuristic measure proposed in previous chapter
- We introduce quantitative guideline and suggestions about using the proposed measure to determine sampling strategies

Data

We utilize four real datasets collected from on-line resources [67, 33, 94, 66] as well as thirty-five synthetic attributed graph datasets to evaluate the proposed methodology.

Real datasets

Cora [67], Citeseer [33], and Pubmed [94] are three citation datasets where scientific publications are nodes, and the references between them form edges. Each paper’s feature is a binary vector indicating the presence of words on the paper. Papers are categorized by corresponding topics (labels).

Amazon-Photo is a subset of the Amazon co-purchase graph [66] where photos sold at Amazon are considered as nodes represented by a vector of keywords in their product review and classified by their product category. The connection between a pair of nodes indicates that they are frequently purchased together.

Dataset	Nodes/Edges/Classes	Description
Cora	2708/5278/7	Scientific publications.
Citeseer	3327/4614/6	Scientific publications.
Pubmed	19717/44325/3	Diabetes-focused scientific publications.
Amazon-Photo	7650/143663/8	Photos sold at Amazon.

Table 4.1: Dataset statistics and domain-specific information.

Synthetic dataset

Thirty five attributed graph datasets are synthesized to imitate scale-free (right-skew degree distribution) networks which are commonly found in practice (Figures 4.1 and 4.2). Each graph contains three clusters (subgraphs) with 100 nodes per cluster. Each subgraph is generated following the Barabási–Albert preferential attachment model [11]. The interconnectivity between a pair of

- On each subgraph, a subset of nodes is chosen using weighted random sampling on degrees of the nodes. We posit that popular nodes (with high degrees) in each subgraph tends to

connect with other popular nodes in other subgraph. In the context of citation network, well-known publications in one class might be cited by popular works in the other classes.

- Random edges are generated between a pair of subsets of nodes. The probability of connecting a pair of nodes is **inter_p**.

The hyperparameters (number of preferential attachment for the Barabási-Albert model, probability of random edges) are then established to control the connectivity between subgraphs (seen in Figures 4.1 and 4.2). As the number of preferential attachment increases, the graph grows in the amount of connectivity and becomes denser. Similarly, as the inter-graph connectivity **inter_p** gets larger, more connection between clusters are generated.

The node feature matrix is generated by first creating a set of three isotropic Gaussian clusters (100 observations per each cluster) in a two-dimensional feature space and then assigning these observations as node features. We control the amount of overlap between three feature clusters by adjusting the distances between cluster centers and the within-cluster standard deviation.

Methodology

Sampling methods

Two procedures of sampling are considered in this study, namely descending and ascending. In descending sampling, training instances are selected by gradually acquiring from the most important nodes to the least important ones. On the contrary, ascending sampling gradually selects training samples starting from the least important nodes to the most important ones.

Three different criteria are used to evaluate the importance (centrality) of a node for sampling orders. In degree sampling, we acquire nodes for training based on their corresponding number

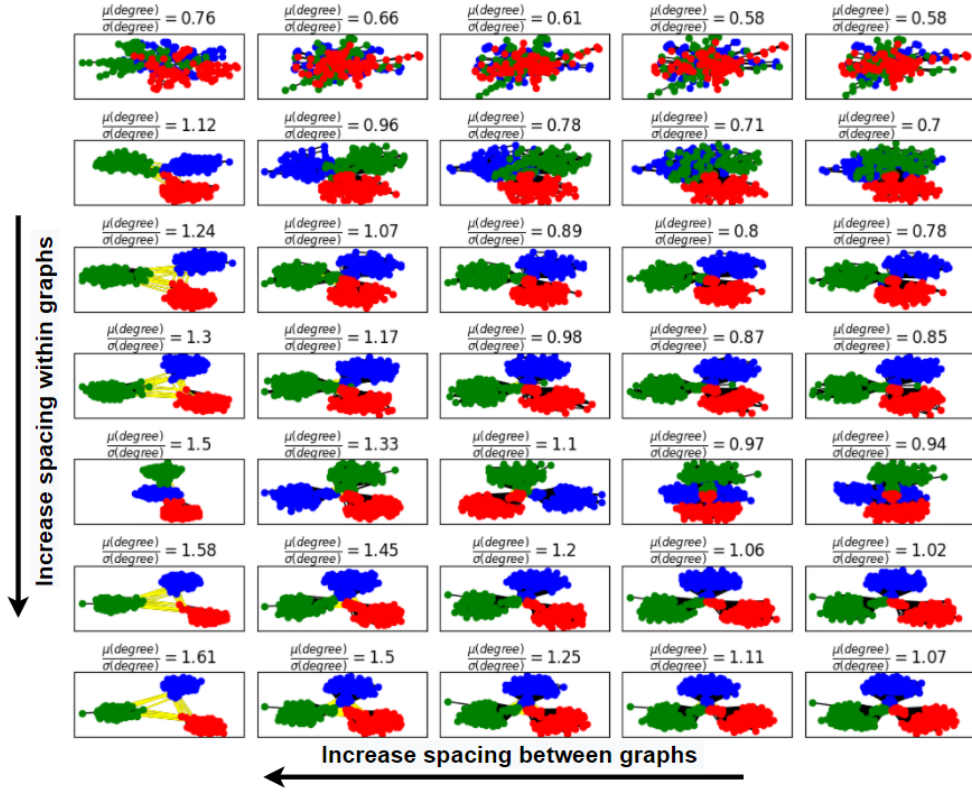


Figure 4.1: Network visualizations for the 35 generated simulations, each with 3 communities (colored). Traversing along the y-axis shows how these networks topologies change when varying the distance within a communities. Traversing along the x-axis shows how the network topologies changes when varying distance between communities.

of directly connected neighbors (i.e. node's degree). The PageRank algorithm [75] derives the rank of a web page (node) by accumulating its incoming neighbors' ranks proportionally to their total number of outgoing connections. The resulting ranking represents the relative importance of pages in the network. In this study, we apply PageRank to rank all nodes in our graphs and then sample them based on their rankings. Lastly, the VoteRank algorithm [128] iteratively selects a set of important nodes called spreaders using voting scores given by the neighboring nodes. Once a node is selected as spreader, it is excluded from next round of voting, and its direct neighbors' voting abilities are also reduced. In this study, we use VoteRank for all nodes on the graph (by

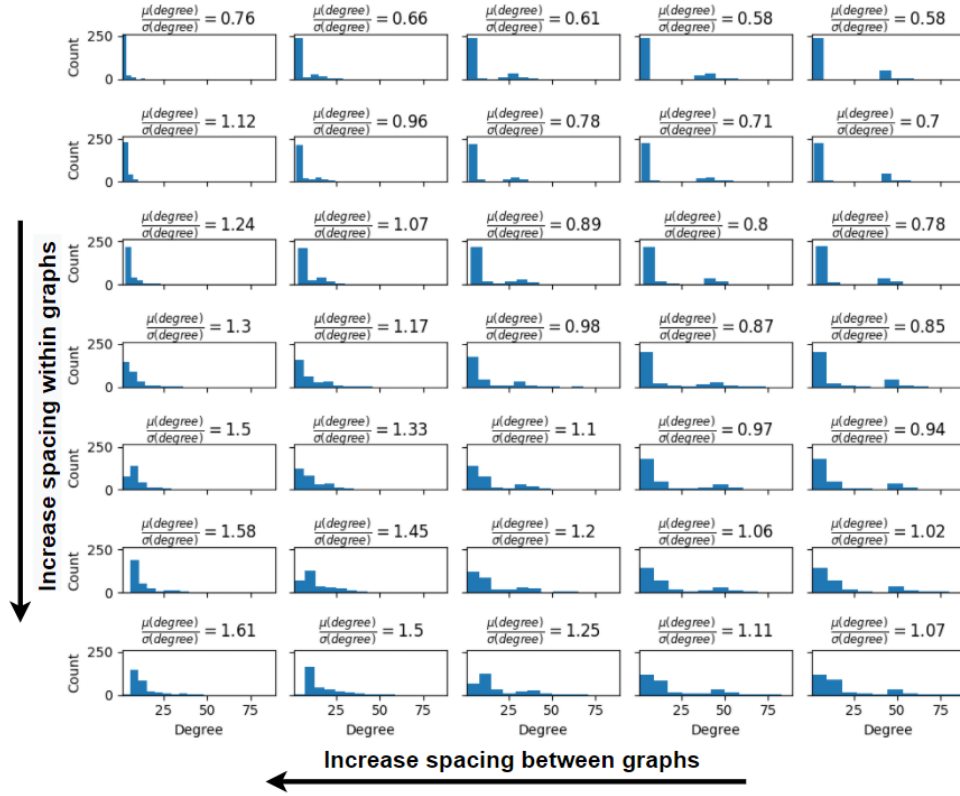


Figure 4.2: Degree distributions for the 35 generated simulations show that all settings create a relatively scale-free network. Traversing along the y-axis shows how these networks topologies change when varying the distance within a communities. Traversing along the x-axis shows how the network topologies changes when varying distance between communities.

setting the number of spreaders as the total number of nodes) and then sample them based on their rankings.

Simple Graph Convolution (SGC)

SGC [121] is a simplified GNN model developed from GCN [48] by removing non-linear activation functions between hidden layers and reparametrizing successive layers into one single layer. This simplification reduces superfluous complexity of GCN while retains superb performance on

many downstream tasks. The work of [79] illustrates SGC’s expressive power in node classification task and proposes a flexible regularization methodology to reduce the number of parameters and highlight a sparse set of important features. The SGC is a ‘one-shot’ learner which simplifies the training procedure and allows for the full set of data points to be used for the parameter inference.

In this section, we briefly present the original SGC. An attributed graph data set contains a graph $G = (V; \mathbf{A})$ and a feature matrix $X \in \mathbb{R}^{N \times D}$. The graph G composes of $V = (v_1, v_2, \dots, v_N)$ is a set of N nodes (vertices) and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix where each element a_{ij} represents an edge between node v_i and v_j ($a_{ij} = 0$ if v_i and v_j are disconnected). We define the degree matrix $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_N)$ as a diagonal matrix whose off-diagonal elements are zero and each diagonal element d_i capture the degree of node v_i and $d_i = \sum_j a_{ij}$. Each row x_i of the feature matrix $X \in \mathbb{R}^{N \times D}$ is the feature vector measured on each node of the graph. Each node i receives a label from C classes and hence can be coded as one hot vector $y_i \in \{0, 1\}^C$.

The GCNs and SGC add self-loops and normalize the adjacency matrix to get the matrix \mathbf{S} :

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \quad (4.1)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \text{diag}(\tilde{\mathbf{A}})$. This normalization allows successive powers of the matrix to not influence the overall size the projections. The SGC removes a nonlinear transformation from the k^{th} layer of the GCN, resulting in a linear model of the form:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S} \dots \mathbf{S} \mathbf{X} \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}). \quad (4.2)$$

The SGC classifier is then achieved by collapsing the repetitive multiplication of matrix \mathbf{S} into the k^{th} power matrix \mathbf{S}^K and reparameterizing the successive weight matrices as $\Theta = \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}$,

and its structure as a GNN is defined by:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S}^K \mathbf{X} \Theta). \quad (4.3)$$

The parameter k corresponds to the number of 'hops' which is the number of edge traversals in the network adjacency matrix \mathbf{S} . k can be thought of as accumulating information from a certain number of hops away from a node (as visually described in [121]). If $k = 0$ the methodology becomes equivalent to a logistic regression application which is known to be scalable to large datasets. Since the SGC introduces the matrix \mathbf{S} as linear operation, the same scalability applies. The weight matrix Θ is trained by minimizing the loss:

$$\mathcal{L} = \sum_{l \in \mathcal{Y}_L} \sum_{c \in C} Y_{lc} \ln \hat{Y}_{lc} \quad (4.4)$$

where \mathcal{Y}_L is a collection of labeled nodes. This model allows for a very computationally efficient exploration of the network based datasets but this multilayer approximation may not provide the full extent of deep learning generalizations.

Evaluation of Network Topology

The network topology is evaluated using the inverse of coefficient of variation of the node's degree distribution.

$$iCV_d = \frac{\mu_d}{\sigma_d} \quad (4.5)$$

where $\mu_d = \frac{1}{N} \sum_{i=1}^N d_i$ is the average degree and $\sigma_d = \frac{1}{N-1} \sum_{i=1}^N (d_i - \mu)^2$ is the standard deviation of degree.

A low value of iCV_d occurs for networks that have a high variation in their node degree distributions compared to the mean degree of the nodes. It indicates that important hubs (nodes) are highly connected to other nodes. On the contrary, a high value of iCV_d results from relatively low variation in degree distribution compared to the mean degree where important nodes tends to be less popular.

The feature information is evaluated using the inverse of coefficient of variation of the node-to-node feature distances. For example, node 1 is defined as a vector of distances between its feature vector and all other nodes' feature vectors. This description allows for an evaluation regarding a node's centrality in the feature space.

Results

In this section, the correlation of the optimal sampling direction for the node classification task with network topology is captured in simulations and real data. The results show that no sampling method (i.e. degree, PageRank, VoteRank) is uniformly superior in terms of accuracy. However, independent of the ascending/descending, we see across the board a higher number of cases where the more complicated sampling procedures (i.e. PageRank/VoteRank) outperform Degree. While we see an increase in performance, there is a trade-off with computation time; nodes degree distribution can be computed swiftly, while PageRank and VoteRank require complex evaluation, and hence, be more computationally expensive.

Dataset	iCV_d	Optimal sampling direction
Cora	0.75	Descending
Citeseer	0.82	Descending
Pubmed	0.6	Ascending
Amazon-Photo	0.69	Ascending

Table 4.2: Optimal sampling results on real datasets

Dataset information including degree inverse of coefficient of variation and optimal sampling direction, as derived through a grid search, can be found in Table 4.2. The descending and ascending optimal sampling directions are cleanly partitioned in the iCV_d space. A numerical boundary would be useful to allow users to calculate iCV_d and perform the active learning procedure without having to experiment through grid search, as done in this paper. From the results (Table 4.2), it is hard to pinpoint an exact threshold, other than that it should likely be somewhere between $iCV_d = 0.69$ and $iCV_d = 0.75$. Therefore, simulations are conducted to obtain a finer resolution. Figure 4.3 shows a contour plot containing the density of the iCV_d distribution for ascending and descending sampling procedures. A partition is found near $iCV_d = 0.82$, which is slightly higher than our estimated window, which is likely caused by discrepancies between the real and simulated data. However, generally speaking, these visualized distributions show significant levels of partitioning on the vertical axis (iCV_d). In fact, a one-sided t-test result concludes significance ($pval = 6.7 \times 10^{-41}$).

In high iCV_d graphs (i.e. Cora and Citeseer), all three descending methods almost uniformly do better than the ascending methods across training sizes (Figure 4.4). Apparent performance improvements are made in terms of accuracy, especially at low train sizes (from $s = 0.1$ to $s = 0.5$). As the training size gets closer to utilizing the full training dataset ($s = 1.0$), sampling approaches are less selective because, by definition, they are using more and more data each iteration. The dominance of descending sampling in these graphs might be explained by the fact that important (central) papers of certain disciplines are usually cited by many papers in that same discipline. Consequently, the most important nodes contain crucial information about the class label and are commonly referenced by papers within its discipline, so they are beneficial for node classification task.

Alternatively, we observe an opposite trend in low iCV_d graphs (Pubmed and amazon-photo), where ascending samplings prevail. The Pubmed citation graph contains publications about a

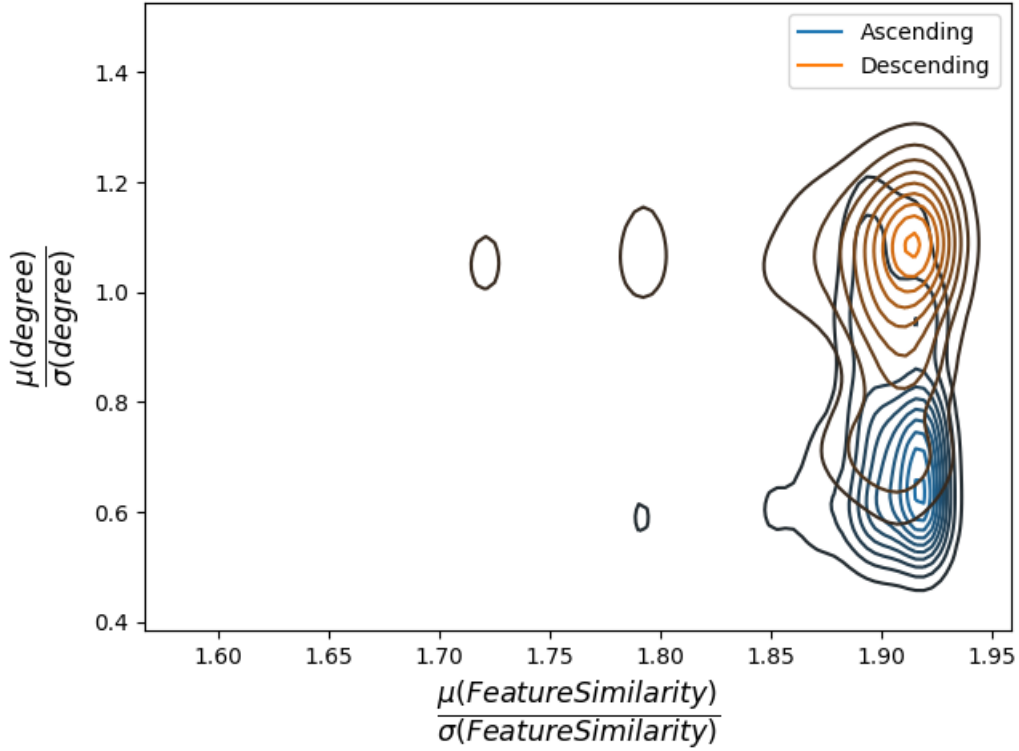


Figure 4.3: Simulations report a density of preferred (higher accuracy) sampling direction as a function of network topology (y-axis) and feature similarity (x-axis) shows that the sampling direction is dependent on the network topology.

specified domain and hence has a smaller scope compared with other citation data sets like Cora and Citeseer (Figure 4.5a). Important (central) papers across classes might cite each other due to the close nature of their categories. Therefore, important nodes contains a less differentiating factor for the classification tasks. On the other hand, less important nodes might contain unique characteristics of the class and render useful information for node classification task. Amazon-photo graph exhibits closely connected clusters with relatively low inter-cluster connectivity (Figure 4.5b). Popular photos from different categories might have similar features (in terms of reviews, as they receive generally positive compliments). Hence, sampling popular nodes is less desirable for classification since their representations are indiscriminative. Less popular photos

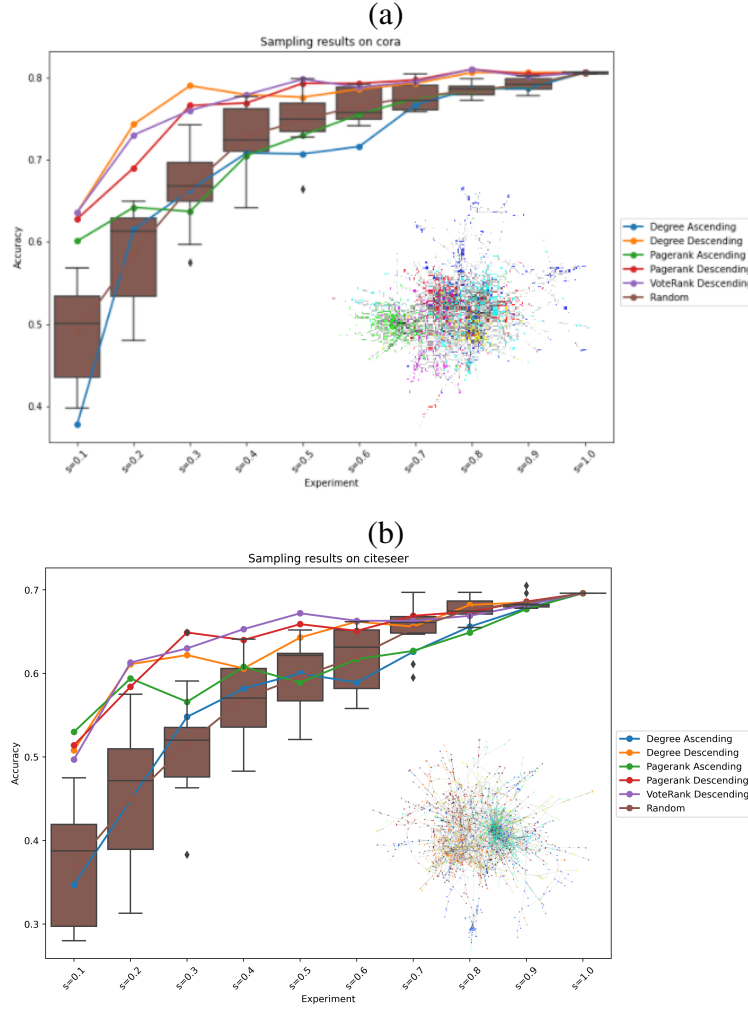


Figure 4.4: High iCV_d networks (Cora, Citeseer) have higher accuracies when sampling nodes from the highest score to lowest score (i.e. 'descending' methods), showing the effectiveness of the node ranking algorithms on a node classification task.

might contain more defined characteristics of their corresponding category. Therefore, lower score nodes may contain more information about the community and hence be more beneficial for node classification.

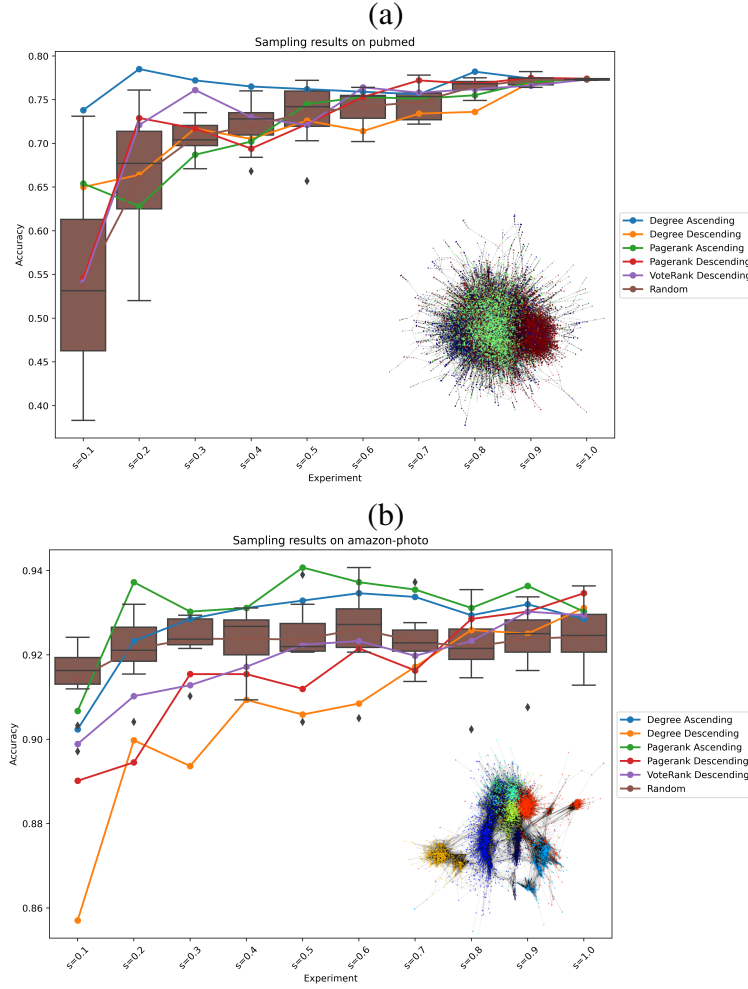


Figure 4.5: Low iCV_d networks (Pubmed and amazon-photo) graphs have lower accuracies when sampling nodes from the lowest to highest score (i.e. 'ascending' methods), showing the ranking algorithms are inversely beneficial to the node classification task.

Conclusions

Participants in networking platforms continue to upload more data onto these platforms, such as in academic literature [124] and social networking [73] (being part of the *always-on* generation [93] and efficient protocols [3]). It becomes a question of efficiency of whether a subset of the nodes can be sampled to provide information about other nodes with unknown membership labels, and

can be useful for e-health [59]. The study conducted here on a set of networks covering different information sources shows that the best indicator for whether nodes should be sampled in terms of ascending or descending centrality is based upon the inverse of coefficient of variation of the degree of the nodes. Intuitively, this can be understood as being related to the sparseness of the network topology. An implication of this is that when attempting to infer labels of network participants in an active learning paradigm, understanding the general degree distribution for communities can determine whether the sampling should be done in the ascending or descending direction. Practitioners can use the general rule of thumb ($iCV_d > 0.8$ should use the descending sampling direction, otherwise ascending) to avoid the computational burden of computing grid searches.

Future work will entail applications in professional networking sites (i.e. LinkedIn) to improve the ability to adopt a community of followers of a certain label, or finding the best connections to develop a new affiliation label. These actions can help navigate the labor market [109] for opportunities or to plan promotions in new markets.

CHAPTER 5: ONE-CLASS GRAPH NEURAL NETWORKS FOR DETECTION OF ANOMALOUS NODES IN ATTRIBUTED GRAPHS (rh-OCGNN)

Introduction

In chapters 2, 3, and 4, we have discussed flexible regularization and efficient sampling strategies for node classification task in attributed graph. In this chapter, we turn our focus to another important task on graph domain, i.e. detection of anomalous nodes, edges, and subgraphs. Studying these uncommon entities helps improve the performance of learning algorithms and the security of the underlying systems. We will present a class of graph neural networks (GNN) that is optimal for anomalous nodes detection on graph structured data.

Anomaly detection involves identifying unusual data that do not follow the expected pattern of the majority within a dataset. Multiple terminologies are devised in different domains to address these irregular data, such as anomalies, outliers, discordant observations, exceptions, etc. [21]. In this work, we use anomaly and outlier interchangeably to refer to these anomalous data points (or nodes). Anomaly detection has captured a large amount of attention in the research community with many applications in various domains. It can be used for administering internet protocol network security measures [112], monitoring crowded scenes in computer vision [63], fraud detection in finance [46], and surveillance of public health care [20]. In those applications, recognizing anomalous patterns plays an essential role, as it provides critical and actionable information for administrative system management or further investigation of irregular events.

Due to its complex nature, anomaly detection still remains a challenging problem. There are many techniques developed for this task which can be categorized as classification-based such as clus-

tering techniques when inferring where to place cluster centers. Interested readers are referred to the works of [21, 80] for a thorough review. A typical approach utilizes regular data to build a model capable of describing the range of a class in the domain of a model. Test data can be scored and compared to a decision threshold (guard value) to determine the points which are considered anomalous [80] or not. Support Vector Data Description (SVDD) [110] is a prominent anomaly detection algorithm aiming to establish a spherical boundary enclosing normal class data. Unknown data will be deemed an anomaly if it resides outside the sphere, i.e. its distance from the sphere’s center exceeds the sphere’s radius.

Attributed graph (network) is a powerful tool to model real-life complex systems where each element is regarded as a node with associated attributes (features) and the connectivity information between elements forms edges. This graph-structured feature data is used to represent complex systems in various domains such as social science (social networks [29]), biology (biochemical pathways [17]), and material science (molecular networks [34]). Research into network-based approaches in machine learning and deep learning has been ongoing with the goal of uncovering information hidden within these data structures. Applications of graph analysis can be categorized into node-level tasks (node classification [85], node regression [74]), edge-level tasks (link prediction [119], edge classification [2]), and graph-level tasks (graph classification [126], graph prediction [125]). A key factor determining the success of those tasks is the effectiveness of the framework in combining two sources of information, i.e. node features and topological structure of the graph, into representative low-dimensional node embedding (in the form of vectors) [99].

Graph Neural Networks (GNNs) are neural networks operating on the graph domain. GNNs produce expressive node representations (node embeddings) via an iterative message passing mechanism where each node aggregates its direct neighbors’ feature vectors and updates its own feature vector with the aggregate information. The final node representation encapsulates structural information of a k-hop neighborhood. To produce a representation of the entire graph, one simply

applies a pooling operator (such as summation) on the set of feature vectors of all nodes [34, 126]. Many GNNs variants have been proposed and achieved state-of-the-art performance on a variety of tasks. Graph Convolutional Networks (GCN) [48] learn a graph representation via layer-wise propagation rules representing localized spectral filters. GraphSAGE [37] generates embeddings by uniformly sampling nodes neighbors and aggregating their features using different choice of aggregators. Graph Attention Network (GAT) [116] utilizes a attention mechanism to account for neighbors' importance in aggregation phase. Simple Graph Convolution (SGC) [121] simplifies GCN by removing non-linear transitions between layers while retaining its representational power. The work of [79] demonstrates the expressive node embedding capability of SGC and explores flexible regularization mechanisms to facilitate meaningful interpretation. Interested readers are referred to [136] for detailed reviews of GNNs.

Graph anomaly detection (GAD) can be defined as the task of identifying uncommon graph entities (nodes, edges, subgraphs) that differ significantly from the expected pattern of reference entities. As graph structured data is gaining popularity due to its power in capturing a rich amount of information, developing an efficient GAD framework becomes necessary. GAD can help in revealing suspicious patterns in networks, and hence provide useful insight to detect fraudulent transactions and strengthen the integrity of complex systems in finance, commercial, telecommunication, etc. [5]. Various GAD methods have been proposed over the years and can be categorized based on the types of anomalous graph entity, the characteristics of input graph data (unattributed or attributed, static or dynamic), availability of annotated training data (supervised, unsupervised, semi-supervised), etc. [81]. In this work, we focus on the detection of anomalous nodes in attributed graph data.

Detecting anomalous nodes in attributed graph data can be achieved by applying traditional machine learning techniques to identify anomalies on the node feature space. This strategy ignores the supplementary node connectivity information and hence often yields poor performance. One

might attempt to incorporate additional features capturing the network topology such as a node’s importance measures (in and out degree, betweenness centrality, closeness centrality, eigenvector centrality, etc. [70]). However, manual feature engineering is often not desirable due to its heavy dependence on human prior knowledge regarding the specific networks and the application domain. On the other hand, network-based techniques [5, 14] utilize only connectivity information to construct densely connected communities in the networks, then mark anomalous nodes as those that do not belong to any community. This approach ignores valuable information provided by node features and fails to detect anomalous patterns that can occur in the feature space. Therefore, it is better to use an approach based upon GNN which utilizes both aspects of the information.

Early GAD works [32, 69, 78] incorporate node feature and connectivity information to discover densely connected communities (in full or subspace of relevant node features) and identify local outliers whose node attributes deviate considerably from direct neighbors. The work of [32] specifies a multivariate probabilistic model on the full set of node attributes, but it does not work well in many real-life applications involving large feature space (due to the curse of dimensionality) or upon arbitrary node feature distributions. A drawback of [69, 78] is its dependence on a good community analysis, which is difficult to perform on highly sparse networks. Additionally, their shallow linear mechanisms are inefficient in capturing complex interactions between node features and their connectivity.

With the recent increase in popularity of GNNs, there are efforts to adopt it for its powerful node representation learning in the task of GAD. The works [25, 58] leverage the graph autoencoder framework with the GCN as an encoder to synthesize and compress node features as well as the graph topology into a low dimensional embedding space. While [25] attempts to detect anomalous nodes via a reconstruction error, [58] accomplishes the detection task via an estimated Gaussian mixture density function on the latent embedding space. As the graph autoencoder objective is to compress and reconstruct original data, the learnt node embedding is not a efficient represen-

tation and is not tailored directly to the GAD task. [120] proposes a One-Class Graph Neural Network (OCGNN) incorporating the SVDD hypersphere learning principle into the GNNs node representation learning mechanism to tackle anomaly detection on attributed graph data. OCGNN offers versatility that allows users to adopt any choice of GNN variants. Its objective function efficiently guides GNNs to explore node embedding space directly tailored to the GAD task, i.e. mapping normal class data compactly around a given center, and at the same time construct a minimal volume hypersphere enclosing the data. The framework uses the sphere’s radius as a decision threshold and detects anomalies as nodes reside outside the sphere’s boundary, i.e. their distances from the sphere’s center exceed the radius. However, since it attempts to learn the radius (decision threshold) directly during the training phase, the model is highly prone to overfitting as the learned threshold might behave sporadically due to ineffective training sampling. Incorporating the radius parameter magnitude into the loss function produces an effect where the ideal radius size can be altered in order to reduce the loss at the detriment of the training and testing classification results. Another drawback is the lack of control over the decision threshold inhibiting users from fine tuning it with respect to their domain applications.

We propose radius as hyperparameter OCGNN (rh-OCGNN) as a superior alternative to OCGNN for the GAD task on attributed graph. Inspired by OCGNN, we modify its design treating the decision threshold radius as a hyperparameter rather than a learnable parameter, which contributes to the loss based upon its magnitude. With the rh-OCGNN, practitioners are able to incorporate domain knowledge in determining appropriate decision threshold. Our design also gives users considerable leeway to fine-tune the decision threshold to optimize their anomaly detection system with respect to any accuracy metrics of interest. Furthermore, the proposed model utilizes training samples more effectively, resulting in an easier tuning process and a more robust decision threshold. The main contributions of this work are listed as follows:

- We demonstrate the pitfall of area under the ROC curve (AUC) as a sole accuracy metric for anomaly detection problem
- We thoroughly investigate the overfitting issue of the state-of-the-art OCGNN, and present is a flexible GAD framework rh-OCGNN combining GNN’s powerful representation learning directly with hypersphere learning for anomaly detection. The rh-OCGNN offers direct control of the sphere’s radius, i.e. the decision threshold, facilitating incorporation of users’ domain knowledge and optimization of anomaly detection system for their applications.

Methodology

Anomaly detection and hypersphere learning

Detecting anomalous nodes in a network can be defined as a one-class classification problem in which a classifier is built upon representative normal class data to describe normal behavior [80]. The model assigns anomaly scores to test data and classifies them as normal or anomalous by comparing their scores to a decision threshold. [110] propose Support Vector Data Description (SVDD), which provides a description of normal class by constructing a compact hypersphere enclosing all (or most of) training normal samples. Anomaly scores of test data are computed based on their distances from the sphere’s center and the decision threshold is assigned as the sphere’s radius.

Given a set of N training samples $x_i \in \mathbb{R}^D, i = 1, 2, \dots, N$ from the normal class and a mapping function $\Phi_F : \mathbb{R}^D \rightarrow \mathbb{R}^F$ to project input data onto a high-dimensional feature space, SVDD aims to find a minimum volume hypersphere characterized by a center $c \in \mathbb{R}^F$ and a radius $R > 0$ to enclose most of the training samples on the feature space \mathbb{R}^F :

$$\begin{aligned}
& \min_{R, c, \xi} R^2 + \tau \sum_{i=1}^N \xi_i \\
& \text{s.t. } \|\Phi_F(x_i) - c\|_2^2 \leq R^2 + \xi_i, \xi_i \geq 0, \forall i
\end{aligned} \tag{5.1}$$

where $\|\cdot\|_2$ is the Euclidean norm and ξ_i are slack variables that allow some relaxation of the set of constraints and hence produce a soft boundary (i.e. some training samples can be outside of the sphere). $\tau > 0$ is a hyperparameter controlling the trade-off between the violations of the boundary and the volume of the sphere. SVDD classifies a test data x_i as an anomaly if it resides outside of the hypersphere, i.e. its distance from the sphere's center exceeds the radius.

Hypersphere learning on attributed graph data

An attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$ consists of three components: a collection of nodes (vertexes) $\mathbf{V} = (v_1, v_2, \dots, v_N)$ containing N nodes, a node attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ where each row $x_{v_i} \in \mathbb{R}^D$ is feature vector of node v_i , and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ where each element a_{ij} is the weighted edge between node v_i and v_j ($a_{ij} = 0$ if v_i and v_j are not connected).

GNNs are powerful neural network frameworks capable of learning representation on attributed graph data [126]. Its power comes from utilizing the message passing mechanism, i.e. iterative aggregating neighborhood representation, to learn expressive node embedding for various downstream tasks such as node classification, link prediction, and graph classification [126, 34, 136]. We denote a L -layer GNN model as $g(\mathbf{X}, \mathbf{A}; \mathcal{W})$ where the input consists of nodes attribute matrix \mathbf{X} and nodes connectivity information (i.e. the adjacency matrix \mathbf{A}). $\mathcal{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L)}\}$ is the set of network weights. The model outputs node embedding $\mathbf{Z} \in \mathbb{R}^{N \times F}$ where each row $z_{v_i} \in \mathbb{R}^F$ is vector representation of node v_i .

A L -layer GNN is effectively a collection of L stacked layers, each of which recursively updates the node representation by aggregating information from 1-hop neighbors. As a result, the model produces node embedding capturing structural information within L -hop neighborhood [126]. For a specific l^{th} layer, the output node embedding $\mathbf{H}^{(l+1)}$ can be derived from the input $\mathbf{H}^{(l)}$ (which is the output node embedding of the previous layer) as follows:

$$\mathbf{H}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{A}, \mathbf{W}^{(l)}) \quad (5.2)$$

[120] propose OCGNN framework for anomaly detection task on graph structured data. Their GAD paradigm combines hypersphere learning objective with GNNs to produce latent node embedding capable of mapping normal class nodes as compactly as possible around a given center. This approach can be seen as an extension of deep one class classification framework proposed in [89] on graph domain. The OCGNN problem can be derived as:

$$\begin{aligned} \min_{r, c, \xi} \quad & r^2 + \frac{1}{\beta K} \sum_{i=1}^K \xi_i \\ \text{s.t.} \quad & \|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 \leq r^2 + \xi_i, \xi_i \geq 0, \forall i \end{aligned} \quad (5.3)$$

The set of constraints on ξ_i can be rewritten as $\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2 \leq \xi_i$ and $\xi_i \geq 0$. This is equivalent to $\xi_i = \max(0, \|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2)$. Thus, the OCGNN problem in 5.3 can be cast as an unconstrained optimization as:

$$\min_{r, c, \mathcal{W}} \quad r^2 + \frac{1}{\beta K} \sum_{v_i \in \mathbf{V}_{\text{tr}}} \max(\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2, 0) \quad (5.4)$$

Formally, OCGNN attempts to minimize the objective loss function:

$$\mathcal{L}(r, c, \mathcal{W}) = \frac{1}{\beta K} \sum_{v_i \in \mathbf{V}_{tr}} \left[\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2 \right]^+ + r^2 \quad (5.5)$$

where $K = |\mathbf{V}_{tr}|$ is the size of the training set $\mathbf{V}_{tr} \subseteq \mathbf{V}$ containing only normal nodes, $[\cdot]^+ = \max(0, \cdot)$ is a max function, $c \in \mathbb{R}^F$ and $r \in \mathbb{R}^+$ are the center and radius of the sphere respectively, and $g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i}$ is the vector representation of node v_i obtained via the chosen GNN $g(\mathbf{X}, \mathbf{A}; \mathcal{W})$. The hyperparameter $\beta \in (0, 1]$ controls the update of the sphere's radius (see the explanation below and Eq. 5.31) and thus determines the flexibility of the model by allowing a fraction of training nodes being mapped outside of the sphere boundary.

The first term of the loss function penalizes the distances of nodes mapped outside of the sphere, while the second term characterizes the goal of hypersphere learning, i.e. minimizing the sphere volume by reducing its radius r .

As OCGNN is developed upon deep neural networks, the minimization of the loss function in Eq. 5.5 can be done using gradient descent algorithm. However, the work of [89] has pointed out that treating the center \mathbf{c} as a free parameter of the optimization would result in a trivial solution. Hence, we adopt the suggestion in [89, 120] to fix the center \mathbf{c} as the mean of the training node representations resulted from an initial forward pass:

$$\mathbf{c} = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} \quad (5.6)$$

Thus, we can consider optimizing the loss functions in Eq. 5.5 with respect to r and \mathcal{W} :

$$\mathcal{L}_R(r, \mathcal{W}) = \frac{1}{\beta K} \sum_{v_i \in \mathbf{V}_{tr}} \left[\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2 \right]^+ + r^2 \quad (5.7)$$

To reduce overfitting, one can impose weight decay regularization placed upon the network weights \mathcal{W} by adding a weight decay term into the loss in Eq. 5.5:

$$\begin{aligned}\mathcal{L}_R(r, \mathcal{W}) &= \frac{1}{\beta K} \sum_{v_i \in \mathbf{V}_{\text{tr}}} \left[\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2 \right]^+ + r^2 + \frac{\lambda}{2} \sum_{l=1}^L \left\| \mathbf{w}^{(l)} \right\|_F^2 \\ \mathcal{L}_R(r, \mathcal{W}) &= \mathcal{L}(r, \mathcal{W}) + \frac{\lambda}{2} \sum_{l=1}^L \left\| \mathbf{w}^{(l)} \right\|_F^2\end{aligned}\tag{5.8}$$

where $\lambda > 0$ is the weight decay coefficient and $\|\cdot\|_F$ denotes the Frobenius norm.

Optimization for OCGNN

In this section, we present the approach to minimize the loss in Eq. 5.8. The general procedure depends on the architecture of the chosen GNN. Here, we discuss how the optimization can be achieved on a framework with 2-layer GNN with P hidden neurals in the first layer and Q output neurals in the second layers. The overall framework is shown in figure 5.1.

For an input attributed graph data consisting of a node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the first layer of the GNN model aggregates neighborhood features and updates the input feature of node v_i as:

$$\bar{\mathbf{h}}_i^{(1)} \leftarrow \frac{1}{d_i + 1} \mathbf{h}_i^{(0)} + \sum_{j=1}^n \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}} \mathbf{h}_j^{(0)}\tag{5.9}$$

or equivalently

$$\bar{\mathbf{h}}_i^{(1)} \leftarrow \frac{1}{d_i + 1} \mathbf{x}_i^{(0)} + \sum_{j=1}^n \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}} \mathbf{x}_j^{(0)}\tag{5.10}$$

This can be expressed compactly as $\bar{\mathbf{H}}^{(1)} = \mathbf{S}\mathbf{H}^{(0)} = \mathbf{S}\mathbf{X}$, where \mathbf{S} is the adjusted adjacency matrix with added self-loops (introduced in Methodology section of chapter 2):

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \quad (5.11)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$.

Then, the weighted input of layer 1 is computed by linearly transforming the input $\bar{\mathbf{h}}^{(1)}$ using a learnable weight matrix $\mathbf{W}^{(1)} \in \mathbb{R}^{P \times D}$:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)} \times \bar{\mathbf{h}}^{(1)} \quad (5.12)$$

The output of layer 1, $h_1 \in \mathbb{R}^P$, is produced by applying an element-wise activation function $\sigma_1(\cdot)$ to the weighted input:

$$\mathbf{h}^{(1)} = \sigma_1(\mathbf{z}^{(1)}) \quad (5.13)$$

Similarly, layer 2 of the GNN model produces the final output of node representation by taking the output of layer 1 as its input and repeating the aforementioned transformations using a learnable weight matrix $\mathbf{W}^{(2)} \in \mathbb{R}^{Q \times P}$:

$$\begin{aligned} \bar{\mathbf{H}}^{(2)} &= \mathbf{S}\mathbf{H}^{(1)} \\ \mathbf{z}^{(2)} &= \mathbf{W}^{(2)} \times \bar{\mathbf{h}}^{(2)} \\ \mathbf{h}^{(2)} &= \sigma_2(\mathbf{z}^{(2)}) \end{aligned} \quad (5.14)$$

Compactly, we can rewrite the above transformation with the 2-layer GNN framework as:

$$\begin{aligned}
\mathbf{H}^{(2)} &= g(\mathbf{X}, \mathbf{A}; \mathcal{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}) \\
&= \sigma_2 \left(\mathbf{S} \mathbf{H}^{(1)} \mathbf{W}^{(2)T} \right) \\
&= \sigma_2 \left(\mathbf{S} \left(\sigma_1 \left[\mathbf{S} \mathbf{X} \mathbf{W}^{(1)T} \right] \right) \mathbf{W}^{(2)T} \right)
\end{aligned} \tag{5.15}$$

Then, with K training samples, the loss of OCGNN can be constructed as:

$$\begin{aligned}
\mathcal{L}_R(r, \mathcal{W}) &= \frac{1}{\beta K} \sum_{k=1}^K \left[\left\| h_k^{(2)} - c \right\|^2 - r^2 \right]^+ + r^2 + \frac{\lambda}{2} \sum_{l=1}^2 \left\| \mathbf{W}^{(l)} \right\|_F^2 \\
\mathcal{L}_R(r, \mathcal{W}) &= \mathcal{L}(r, \mathcal{W}) + \frac{\lambda}{2} \sum_{l=1}^2 \left\| \mathbf{W}^{(l)} \right\|_F^2
\end{aligned} \tag{5.16}$$

To minimize the loss in Eq. 5.16, we utilize the gradient descent algorithm, which requires the gradient of the parameters $r, \mathbf{W}^{(1)}, \mathbf{W}^{(2)}$:

The gradient w.r.t. radius r is as follows:

$$\frac{\partial \mathcal{L}_R}{\partial r} = 2r + \frac{1}{\beta K} \sum_{i=1}^K \begin{cases} 0 & \text{if } \|h_i^{(2)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ -2r & \text{else} \end{cases} \tag{5.17}$$

To compute the gradient w.r.t. $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, we apply backpropagation algorithm. Let $\mathcal{L}_i = \frac{1}{\beta} \left[\left\| h_i^{(2)} - c \right\|^2 - r^2 \right]^+$ be the loss of one training sample i . We first compute the gradient of L_i w.r.t. $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ using the following procedure:

1. Compute the gradient of weighted input of layer 2:

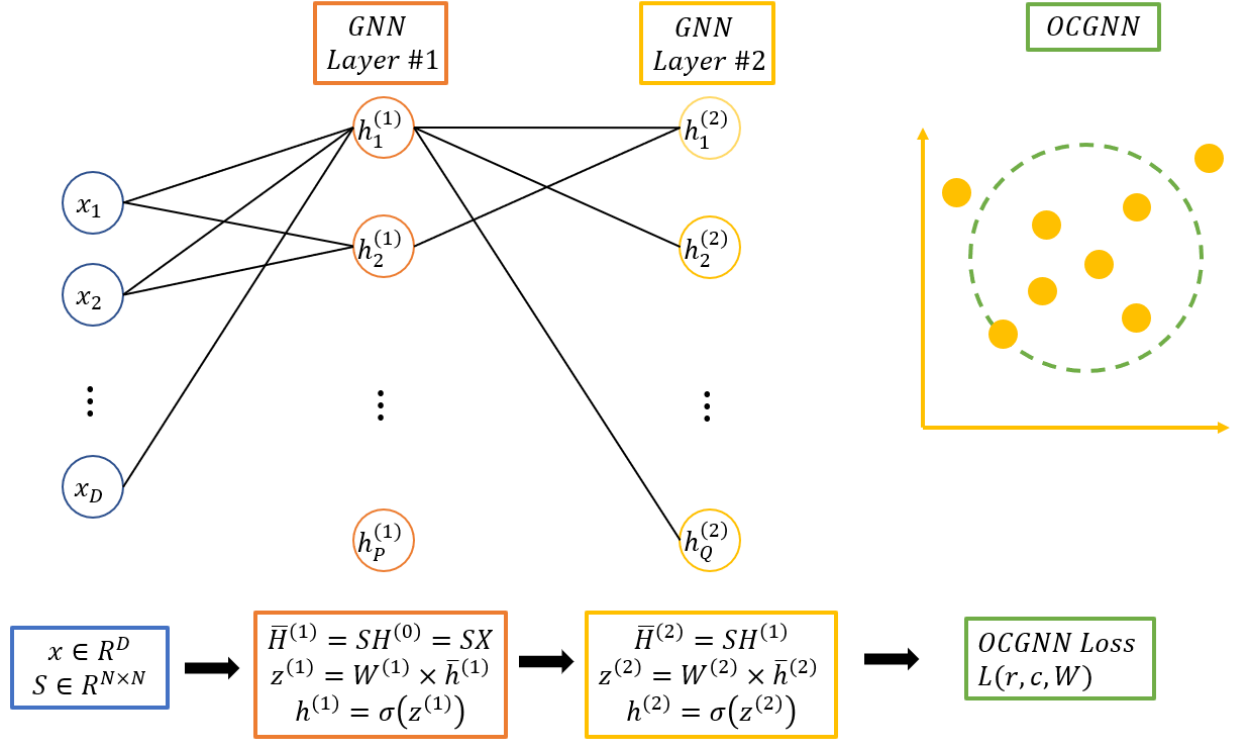


Figure 5.1: The overall framework of GAD involving a 2-layer GNN. The GNN maps the original feature space $x \in R^D$ to the node embedding $h^{(2)} \in R^Q$

$$\delta^{(2)} = \frac{\partial \mathcal{L}_i}{\partial z^{(2)}} = \frac{\partial \mathcal{L}_i}{\partial h^{(2)}} \cdot \frac{\partial h^{(2)}}{\partial z^{(2)}} \quad (5.18)$$

We have:

$$\frac{\partial \mathcal{L}_i}{\partial h^{(2)}} = \begin{cases} 0 & \text{if } \|h^{(2)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ \frac{1}{\beta}(h^{(2)} - \mathbf{c}) & \text{else} \end{cases} \quad (5.19)$$

$$\frac{\partial h^{(2)}}{\partial z^{(2)}} = \sigma'_2(z^{(2)}) \quad (5.20)$$

Hence:

$$\delta^{(2)} = \left(\begin{cases} 0 & \text{if } \|h^{(2)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ \frac{1}{\beta}(h^{(2)} - \mathbf{c}) & \text{else} \end{cases} \right) \odot \sigma'_2(z^{(2)}) \quad (5.21)$$

2. Compute the gradient of input of layer 2:

$$\bar{\delta}^{(2)} = \frac{\partial \mathcal{L}_i}{\partial \bar{h}^{(2)}} = \frac{\partial \mathcal{L}_i}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial \bar{h}^{(2)}} = \mathbf{W}^{(2)T} \cdot \delta^{(2)} \quad (5.22)$$

3. Compute the gradient of output of layer 1:

$$\gamma^{(1)} = \frac{\partial \mathcal{L}_i}{\partial h_i^{(1)}} = \sum_{j \in \mathcal{N}_i} \frac{\partial \mathcal{L}_i}{\partial \bar{h}_j^{(2)}} \cdot \frac{\partial \bar{h}_j^{(2)}}{\partial h_i^{(1)}} = \sum_{j \in \mathcal{N}_i} \bar{\delta}_j^{(2)} \cdot \frac{\partial \bar{h}_j^{(2)}}{\partial h_i^{(1)}} \quad (5.23)$$

We have:

$$\frac{\partial \bar{h}_j^{(2)}}{\partial h_i^{(1)}} = \begin{cases} \mathbf{diag}\left(\frac{1}{d_k+1}\right) & \text{if } i = j \\ \mathbf{diag}\left(\frac{a_{ji}}{\sqrt{(d_i+1)(d_j+1)}}\right) & \text{else} \end{cases} \quad (5.24)$$

where \mathcal{N}_i contains sample (node) i and its 1-hop neighboring samples (nodes) in the training data and $\mathbf{diag}(\cdot)$ returns diagonal matrix of size $P \times P$

4. Compute the gradient of weighted input of layer 1:

$$\delta^{(1)} = \frac{\partial \mathcal{L}_i}{\partial z^{(1)}} = \frac{\partial \mathcal{L}_i}{\partial h^{(1)}} \cdot \frac{\partial h^{(1)}}{\partial z^{(1)}} = \gamma^{(1)} \odot \sigma'_1(z^{(1)}) \quad (5.25)$$

5. Compute the gradient of $\mathbf{W}^{(2)}$:

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathcal{L}_i}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial \mathbf{W}^{(2)}} = \delta^{(2)T} \cdot \bar{h}^{(2)} \quad (5.26)$$

6. Compute the gradient of $\mathbf{W}^{(1)}$:

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathcal{L}_i}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial \mathbf{W}^{(1)}} = \delta^{(1)T} \bar{h}^{(1)} \quad (5.27)$$

Then, the gradient of the loss in Eq. 5.16 w.r.t. $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ can be achieved as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} = \frac{1}{K} \sum_{i=1}^K \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(1)}} + \lambda \cdot \mathbf{W}^{(1)} = \frac{1}{K} \sum_{i=1}^K \delta_i^{(1)T} \bar{h}_i^{(1)} + \lambda \cdot \mathbf{W}^{(1)} \quad (5.28)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} = \frac{1}{K} \sum_{i=1}^K \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(2)}} + \lambda \cdot \mathbf{W}^{(2)} = \frac{1}{K} \sum_{i=1}^K \delta_i^{(2)T} \bar{h}_i^{(2)} + \lambda \cdot \mathbf{W}^{(2)} \quad (5.29)$$

Now, to minimize the OCGNN loss in Eq. 5.16, we iteratively update the parameters r , $\mathbf{W}^{(1)}$, and $\mathbf{W}^{(2)}$ with gradient descent algorithm.

$$\begin{aligned} r^{(t+1)} &\leftarrow r^{(t)} - \eta_r \times \left[2r + \frac{1}{\beta K} \sum_{i=1}^K \begin{cases} 0 & \text{if } \|h^{(2)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ -2r & \text{else} \end{cases} \right] \\ \mathbf{W}^{(1)(t+1)} &\leftarrow \mathbf{W}^{(1)(t)} - \eta_W \frac{1}{K} \sum_{i=1}^K \delta_i^{(1)T} \bar{h}_i^{(1)} + \lambda \cdot \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)(t+1)} &\leftarrow \mathbf{W}^{(2)(t)} \eta_W \frac{1}{K} \sum_{i=1}^K \delta_i^{(2)T} \bar{h}_i^{(2)} + \lambda \cdot \mathbf{W}^{(2)} \end{aligned} \quad (5.30)$$

where η_r and η_W are step sizes for the update.

As suggested by the work of [120], we update r and \mathcal{W} alternatively since r is not an inner parameter of the GNN model. In particular, we first fix r and update \mathcal{W} only. Then, after every ϕ^{th} iteration, with the latest update \mathcal{W}' , we compute the distance set $\mathbf{d}_{\mathbf{V}_{tr}}$ of the training samples and r

can be updated by linear percentile search on the distance set:

$$\begin{aligned}
\mathbf{d}\mathbf{v}_{tr} &= \{d_{v_i} | d_{v_i} = \|g(\mathbf{X}, \mathbf{A}; \mathcal{W}')_{v_i} - \mathbf{c}\|^2, v_i \in \mathbf{V}_{tr}\} \\
&= \{d_{v_i} | d_{v_i} = \|h_{v_i}^{(2)} - \mathbf{c}\|^2, v_i \in \mathbf{V}_{tr}\} \\
r &\leftarrow P_{(1-\beta) \times 100\%}(\mathbf{d}\mathbf{v}_{tr})
\end{aligned} \tag{5.31}$$

where $P_i\%$ is the i -th percentile. It's obvious that after a round of update, there are $(\beta) \times 100\%$ of training nodes mapped outside of the boundary since their distances exceed r .

The backpropagation algorithm to update a general L -layer OCGNN can be summarized as follows:

1. Compute the gradient of weighted input of layer L :

$$\begin{aligned}
\delta^{(L)} &= \frac{\partial \mathcal{L}_i}{\partial \mathbf{z}^{(L)}} = \frac{\partial \mathcal{L}_i}{\partial \mathbf{h}^{(L)}} \cdot \frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{z}^{(L)}} \\
&= \left(\begin{cases} 0 & \text{if } \|\mathbf{h}^{(L)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ \frac{1}{\beta}(\mathbf{h}^{(L)} - \mathbf{c}) & \text{else} \end{cases} \right) \odot \sigma'_L(\mathbf{z}^{(L)})
\end{aligned} \tag{5.32}$$

2. Backpropagate the gradient:

For each $l = L - 1, L - 2, \dots, 1$ compute the gradient of weighted input of layer l

$$\delta^{(l)} = \frac{\partial \mathcal{L}_i}{\partial \mathbf{z}^{(l)}} = \frac{\partial \mathcal{L}_i}{\partial \mathbf{h}^{(l)}} \cdot \frac{\partial \mathbf{h}^{(l)}}{\partial \mathbf{z}^{(l)}} = \gamma^{(l)} \odot \sigma'_l(\mathbf{z}^{(l)}) \tag{5.33}$$

where:

$$\bar{\delta}^{(l+1)} = \frac{\partial \mathcal{L}_i}{\partial \bar{\mathbf{h}}^{(l+1)}} = \frac{\partial \mathcal{L}_i}{\partial \mathbf{z}^{(l+1)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \bar{\mathbf{h}}^{(l+1)}} = \mathbf{W}^{(l+1)T} \cdot \delta^{(l+1)} \tag{5.34}$$

$$\gamma^{(l)} = \frac{\partial \mathcal{L}_i}{\partial h_i^{(l)}} = \sum_{j \in \mathcal{N}_i} \frac{\partial \mathcal{L}_i}{\partial \bar{h}_j^{(l+1)}} \cdot \frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} = \sum_{j \in \mathcal{N}_i} \bar{\delta}_j^{(l+1)} \cdot \frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} \quad (5.35)$$

$$\frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} = \begin{cases} \mathbf{diag}\left(\frac{1}{d_{k+1}}\right) & \text{if } i = j \\ \mathbf{diag}\left(\frac{a_{ji}}{\sqrt{(d_i+1)(d_j+1)}}\right) & \text{else} \end{cases} \quad (5.36)$$

where \mathcal{N}_i contains sample (node) i and its 1-hop neighboring samples (nodes) in the training data and $\mathbf{diag}(\cdot)$ returns diagonal matrix

3. Compute the gradient w.r.t. weight $\mathbf{W}^{(l)}$ of layer l :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{1}{K} \sum_{i=1}^K \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} + \lambda \cdot \mathbf{W}^{(l)} = \frac{1}{K} \sum_{i=1}^K \delta_i^{(l)T} \cdot \bar{h}_i^{(l)} + \lambda \cdot \mathbf{W}^{(l)} \quad (5.37)$$

Training OCGNN

The training procedure of OCGNN can be summarized in Algorithm 3. At the outset, the set of weights \mathcal{W} is initialized and used to produce the initial node representation matrix $\mathbf{H}^{(L)} = g(\mathbf{X}, \mathbf{A}; \mathcal{W}) \in \mathbb{R}^{N \times F}$. The radius r is set as 0, and the center is computed as the mean of node representation in training data $c = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} h_{v_i}^{(L)}$. Then, the set of distances between the center and training nodes are computed $\mathbf{d}_{\mathbf{V}_{tr}} = \{d_{v_i} | d_{v_i} = \|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2, v_i \in \mathbf{V}_{tr}\}$ and the loss is derived as in Eq. 5.8. The framework can optimize the loss by updating the network weights \mathcal{W} using stochastic gradient descent while the radius r is updated alternately (after every ϕ^{th} epochs during training phase) via linear percentile search on the distance set $\mathbf{d}_{\mathbf{V}_{tr}}$ as follows:

$$r \leftarrow P_{(1-\beta) \times 100\%}(\mathbf{d}_{\mathbf{V}_{tr}}) \quad (5.38)$$

Note that in this work, we update neural networks weights \mathcal{W} using PyTorch [77] which is an open source deep learning framework.

Algorithm 3 Training OCGNN models

Input: Attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$, normal training nodes \mathbf{V}_{tr} , β

Output: Weight $\mathcal{W} = \{\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}\}$, radius $r \in \mathbb{R}^+$, and center $c \in \mathbb{R}^F$

```

1: Initialize  $\mathcal{W}^{[i=0]}$ 
2: Initialize  $r^{[i=0]} = 0$ ; Compute  $\mathbf{c} = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i=0]})_{v_i}$ 
3: while  $i \leq \text{num\_iter}$  do
4:   Compute node representation:  $H^{(L)[i]} = g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i]})$ 
5:   Compute distance set:  $\mathbf{d}_{\mathbf{V}_{tr}}^{[i]} = \{d_{v_i} | d_{v_i} = \|h_{v_i}^{[i](L)} - \mathbf{c}\|^2, v_i \in \mathbf{V}_{tr}\}$ 
6:   Compute the loss:  $\mathcal{L}_R^{[i]} = \frac{1}{\beta K} \sum_{k=1}^K \left[ \mathbf{d}_{\mathbf{V}_{tr}}^{[i]} - (r^{[i]})^2 \right]^+ + (r^{[i]})^2 + \frac{\lambda}{2} \sum_{l=1}^L \|\mathbf{W}^{[i](l)}\|_F^2$ 
7:   Update  $\mathcal{W}^{[i+1]}$  using gradient descent and backpropagation algorithms
8:   if  $i \bmod \phi = 0$  then
9:     Update radius:  $r^{[i+1]} \leftarrow P_{(1-\beta) \times 100\%}(\mathbf{d}_{\mathbf{V}_{tr}}^{[i]})$ 
10:  end if
11:   $i \leftarrow i + 1$ 
12: end while

```

After training phase, the optimal \mathcal{W}^* and r^* can be used to calculate the distance from the center $d_{v_i} = \|g(\mathbf{X}, \mathbf{A}; \mathcal{W}^*)_{v_i} - \mathbf{c}\|^2$ and anomaly score $S(v_i) = d_{v_i} - r^{*2}$ for a node $v_i \in \mathbf{V}$. Node v_i is then classified as an anomaly if $S(v_i) > 0$ or a normal node otherwise.

rh-OCGNN

The above OCGNN framework detects anomalous nodes by comparing their distances from the sphere center (on the GNN’s latent embedding space) and the radius. In effect, the radius is an intrinsic threshold of the model. Tuning threshold is important to achieve decent performance for anomaly detection task [108, 117]. One drawback of the above OCGNN is the incapability of tuning the radius directly. Since the radius is learned to optimize the loss (Eq. 5.5) using training data, the boundary achieved might be overfitting and not generalize well on test data. Although

one might control the volume of the sphere via β , Eq. 5.31 suggests that when β is set too small, the linear percentile search will result in too few training samples mapped outside the boundary. The consequence is less training data fed into the penalty part of the loss function (the first term in Eq. 5.5) inhibiting the ability to learn a good embedding space to describe normal class data.

Instead of building a hypersphere with the smallest volume in the latent space of GNN, we encourage the model to learn a mapping space (via the set of weights \mathcal{W}) that projects normal class data compactly within a hypersphere of given radius. Figure 5.2 illustrates the architecture of the proposed framework. This is equivalent to the following optimization problem:

$$\begin{aligned} \min_{c, \xi} \quad & \frac{1}{K} \sum_{i=1}^K \xi_i \\ \text{s.t.} \quad & \|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 \leq r^2 + \xi_i, \xi_i \geq 0, \forall i \end{aligned} \quad (5.39)$$

With similar derivation as described earlier, the problem in 5.39 can be cast as an unconstrained optimization as:

$$\min_{c, \mathcal{W}} \frac{1}{K} \sum_{v_i \in \mathbf{V}_{\text{tr}}} \max(\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2) \quad (5.40)$$

Formally, our proposed framework aims to minimize the following loss function:

$$\mathcal{L}(\mathcal{W}) = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{\text{tr}}} \left[\|g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} - c\|^2 - r^2 \right]^+ + \frac{\lambda}{2} \sum_{l=1}^L \left\| \mathbf{W}^{(l)} \right\|_F^2 \quad (5.41)$$

Again, we adopt the suggestion from the work of [89, 120] to fix the hypersphere center c as the

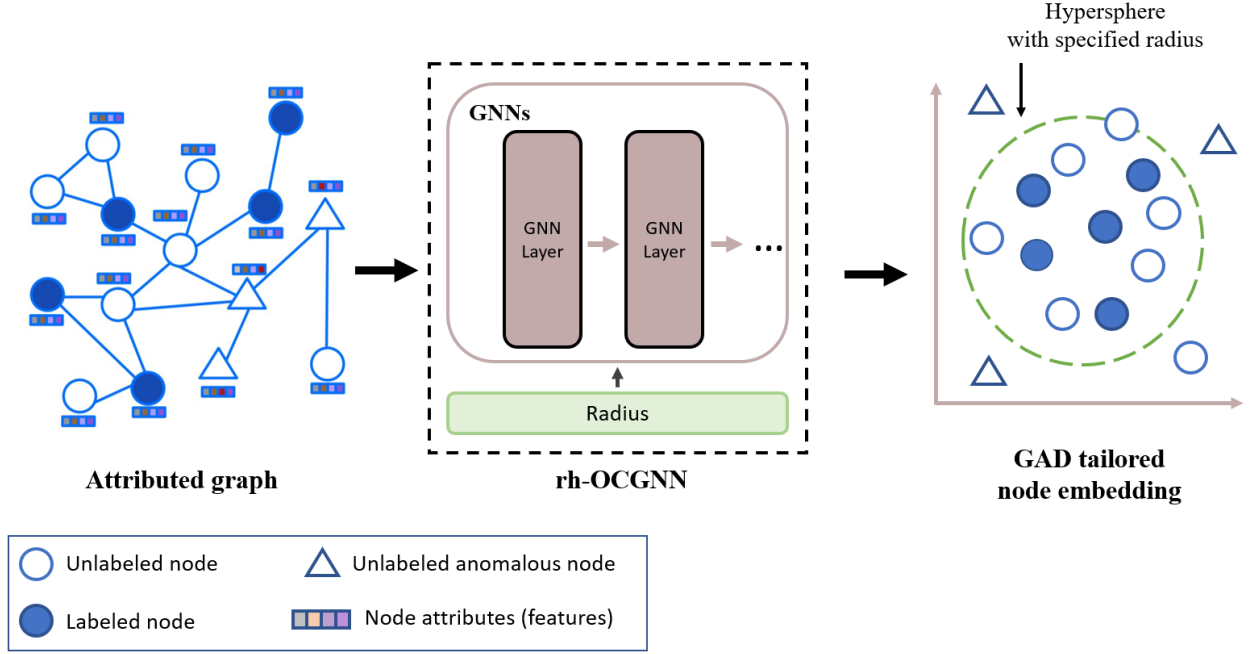


Figure 5.2: The overall framework of our proposed rh-OCGNN for graph anomaly detection (GAD) on attributed graph data. Given an input attributed graph, rh-OCGNN guides GNNs to explore embedding space mapping normal nodes closely within a hypersphere of user-specified radius r .

mean of the training node representations resulting from an initial forward pass:

$$\mathbf{c} = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{\text{tr}}} g(\mathbf{X}, \mathbf{A}; \mathcal{W})_{v_i} \quad (5.42)$$

In Eq. 5.41, r becomes a hyperparameter determined by the user before training the model. The first term is a penalization of the distances of nodes mapped outside of the sphere and the second term is weight decay regularization. To distinguish from the above OCGNN, we name our model as "radius as hyperparameter OCGNN" (rh-OCGNN). Our rh-OCGNN framework allows practitioners to employ domain knowledge in choosing appropriate decision threshold value (in form of the radius) for their anomaly detection applications. In addition, users can further optimize their

anomaly detector by directly tuning the radius with respect to any performance metrics related to the corresponding domains.

Optimization for rh-OCGNN

To minimize the loss in Eq. 5.41, we apply the same backpropagation algorithm described above to derive the gradient of the weights $\mathcal{W} = \mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}$, then utilize gradient descent to update them, respectively, as follows:

1. Compute the gradient of weighted input of layer L:

$$\begin{aligned} \delta^{(L)} &= \frac{\partial \mathcal{L}_i}{\partial z^{(L)}} = \frac{\partial \mathcal{L}_i}{\partial h^{(L)}} \cdot \frac{\partial h^{(L)}}{\partial z^{(L)}} \\ &= \left(\begin{cases} 0 & \text{if } \|h^{(L)} - \mathbf{c}\|^2 - r^2 \leq 0 \\ \frac{1}{\beta}(h^{(L)} - \mathbf{c}) & \text{else} \end{cases} \right) \odot \sigma'_L(z^{(L)}) \end{aligned} \quad (5.43)$$

2. Backpropagate the gradient:

For each $l = L - 1, L - 2, \dots, 1$ compute the gradient of weighted input of layer l

$$\delta^{(l)} = \frac{\partial \mathcal{L}_i}{\partial z^{(l)}} = \frac{\partial \mathcal{L}_i}{\partial h^{(l)}} \cdot \frac{\partial h^{(l)}}{\partial z^{(l)}} = \gamma^{(l)} \odot \sigma'_l(z^{(l)}) \quad (5.44)$$

where:

$$\bar{\delta}^{(l+1)} = \frac{\partial \mathcal{L}_i}{\partial \bar{h}^{(l+1)}} = \frac{\partial \mathcal{L}_i}{\partial z^{(l+1)}} \cdot \frac{\partial z^{(l+1)}}{\partial \bar{h}^{(l+1)}} = \mathbf{W}^{(l+1)T} \cdot \delta^{(l+1)} \quad (5.45)$$

$$\gamma^{(l)} = \frac{\partial \mathcal{L}_i}{\partial h_i^{(l)}} = \sum_{j \in \mathcal{N}_i} \frac{\partial \mathcal{L}_i}{\partial \bar{h}_j^{(l+1)}} \cdot \frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} = \sum_{j \in \mathcal{N}_i} \bar{\delta}_j^{(l+1)} \cdot \frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} \quad (5.46)$$

$$\frac{\partial \bar{h}_j^{(l+1)}}{\partial h_i^{(l)}} = \begin{cases} \mathbf{diag}\left(\frac{1}{d_k+1}\right) & \text{if } i = j \\ \mathbf{diag}\left(\frac{a_{ji}}{\sqrt{(d_i+1)(d_j+1)}}\right) & \text{else} \end{cases} \quad (5.47)$$

where \mathcal{N}_i contains sample (node) i and its 1-hop neighboring samples (nodes) in the training data and $\mathbf{diag}(\cdot)$ returns diagonal matrix

3. Compute the gradient w.r.t. weight $\mathbf{W}^{(l)}$ of layer l :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{1}{K} \sum_{i=1}^K \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} + \lambda \cdot \mathbf{W}^{(l)} = \frac{1}{K} \sum_{i=1}^K \delta_i^{(l)T} \cdot \bar{h}_i^{(l)} + \lambda \cdot \mathbf{W}^{(l)} \quad (5.48)$$

where η_W are step sizes for the update.

Training rh-OCGNN

The training procedure of rh-OCGNN can be summarized in Algorithm 4. At the outset, the set of weights is initialized and used to produce an initial node representation matrix $\mathbf{H}^{(L)} = g(\mathbf{X}, \mathbf{A}; \mathcal{W}) \in \mathbb{R}^{N \times F}$. The center is set as the mean of initial node representation $c = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} h_{v_i}^{(L)}$. Then, the set of distances between the center and training nodes are computed $\mathbf{d}_{\mathbf{V}_{tr}} = \{d_{v_i} | d_{v_i} = \|h_{v_i}^{(L)} - c\|^2, v_i \in \mathbf{V}_{tr}\}$ and the loss is derived as in Eq. 5.41. The framework can optimize the loss by updating the network weights \mathcal{W} using stochastic gradient descent. Again, we utilize PyTorch [77] to perform updating the network's weights \mathcal{W} .

After the training phase, the optimal \mathcal{W}^* can be used to calculate the distance from the center $d_{v_i} = \|g(\mathbf{X}, \mathbf{A}; \mathcal{W}^*)_{v_i} - c\|^2$ and the anomaly score $S(v_i) = d_{v_i} - r^2$ for a node $v_i \in \mathbf{V}$. Node v_i is then classified as an anomaly if $S(v_i) > 0$ or as a normal node otherwise.

Algorithm 4 Training rh-OCGNN models

Input: Attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$, normal training nodes \mathbf{V}_{tr} , radius $r \in \mathbb{R}^+$

Output: Weight $\mathcal{W} = \{\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}\}$, center $c \in \mathbb{R}^F$

- 1: Initialize $\mathcal{W}^{[i=0]}$
 - 2: Compute $\mathbf{c} = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i=0]})_{v_i}$
 - 3: **while** $i \leq \text{num_iter}$ **do**
 - 4: Compute node representation: $H^{[i](L)} = g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i]})$
 - 5: Compute distance set: $\mathbf{d}_{\mathbf{V}_{tr}}^{[i]} = \{d_{v_i} | d_{v_i} = \left\| h_{v_i}^{[i](L)} - \mathbf{c} \right\|^2, v_i \in \mathbf{V}_{tr}\}$
 - 6: Compute the loss: $\mathcal{L}_R^{[i]} = \frac{1}{\beta K} \sum_{k=1}^K \left[\mathbf{d}_{\mathbf{V}_{tr}}^{[i]} - r^2 \right]^+ + \frac{\lambda}{2} \sum_{l=1}^L \left\| \mathbf{W}^{[i](l)} \right\|_F^2$
 - 7: Update $\mathcal{W}^{[i+1]}$ using gradient descent and backpropagation algorithms
 - 8: $i \leftarrow i + 1$
 - 9: **end while**
-

Tuning the radius

One of the important hyperparameters in our rh-OCGNN is the radius r . As described above, the radius r is the threshold of anomaly system. With our framework, users have the flexibility to specify the value of r based on their domain knowledge and applications.

In some applications, without prior knowledge or domain requirement for the threshold r , users have the option to perform hyperparameter optimization to achieve the best performance. Here, we describe a grid search process to tune the radius r in Algorithm 5. First, the normal samples are partitioned into training set \mathbf{V}_{tr} and validation set \mathbf{V}_{val} . Then we choose a performance metric \mathcal{M} to evaluate the capability of the model and set of candidate radii $arr_r = \{r_0, r_1, \dots, r_m\}$ of interest. For each values of radius $r = r_i$ belonging to the candidate set, we train rh-OCGNN on the training set and store its performance on the validation set \mathcal{M}_{r_i} . Depending on the application, the optimal radius r^* is chosen as one that maximizes or minimizes the corresponding set of performance metric:

$$r^* = \min_{r_i} \{\mathcal{M}_{r_1}, \mathcal{M}_{r_2}, \dots, \mathcal{M}_{r_m}\}$$

OR (5.49)

$$r^* = \max_{r_i} \{\mathcal{M}_{r_1}, \mathcal{M}_{r_2}, \dots, \mathcal{M}_{r_m}\}$$

Algorithm 5 Tuning hyperparameter r for rh-OCGNN models

Input: Attributed graph $G = (\mathbf{V}; \mathbf{X}; \mathbf{A})$, normal training nodes \mathbf{V}_{tr} , validation nodes \mathbf{V}_{val} , array of radii $arr_r = \{r_0, r_1, \dots, r_m\}$

Output: Optimal radius r^* and Weight \mathcal{W}^* , center $c \in \mathbb{R}^F$

```

1: for  $r_i$  in  $arr_r$  do
2:   Assign the radius  $r = r_i$ 
3:   Initialize  $\mathcal{W}^{[i=0]}$ 
4:   Compute  $\mathbf{c} = \frac{1}{K} \sum_{v_i \in \mathbf{V}_{tr}} g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i=0]})_{v_i}$ 
5:   while  $i \leq \text{num\_iter}$  do
6:     Compute node representation:  $H^{[i](L)} = g(\mathbf{X}, \mathbf{A}; \mathcal{W}^{[i]})$ 
7:     Compute distance set:  $\mathbf{d}_{\mathbf{V}_{tr}}^{[i]} = \{d_{v_i} | d_{v_i} = \|h_{v_i}^{[i](L)} - c\|^2, v_i \in \mathbf{V}_{tr}\}$ 
8:     Compute the loss:  $\mathcal{L}_R^{[i]} = \frac{1}{\beta K} \sum_{k=1}^K [\mathbf{d}_{\mathbf{V}_{tr}}^{[i]} - r^2]^+ + \frac{\lambda}{2} \sum_{l=1}^L \|\mathbf{W}^{[i](l)}\|_F^2$ 
9:     Update  $\mathcal{W}^{[i+1]}$  using gradient descent and backpropagation algorithms
10:     $i \leftarrow i + 1$ 
11:   end while
12:   Check model's performance on validation set and store  $\mathcal{M}_{r_i}$ 
13: end for
14:  $r^* = \min_{r_i} \{\mathcal{M}_{r_1}, \mathcal{M}_{r_2}, \dots, \mathcal{M}_{r_m}\}$  OR  $r^* = \max_{r_i} \{\mathcal{M}_{r_1}, \mathcal{M}_{r_2}, \dots, \mathcal{M}_{r_m}\}$ 

```

Data

Three popular and publicly available attributed graph datasets are employed in order to explore the proposed model, namely Cora [67], Citeseer [33], and Pubmed [94]. These datasets contain scientific publications (regarded as nodes) where each paper is represented by a binary vector indicating the presence of informative keywords (regarded as node's features). The citations between papers

form edges and each paper can be categorized into a specific topic (regarded as label). Datasets statistics are given in Table 6.1.

We follow the processing strategy in [120] to derive suitable data for the GAD task, that is, assigning one category as normal class (“Neural Networks”, “IR” and “Diabetes Mellitus Type 2” for Cora, Citeseer, and Pubmed respectively) and the remaining categories as abnormal class. The training set contains randomly sampled nodes from the normal class only. On the contrary, half of the validation and test sets consist of normal nodes and the other half are of abnormal nodes. The proportion of normal samples in the train, validation, and test sets are 60/15/25% of total normal class data, respectively.

Dataset	Nodes/Edges/Classes	Features	Train/Val/Test
Cora	2708/5429/7	1433	490/246/410
Citeseer	3327/4732/6	3703	420/210/352
Pubmed	19717/44338/3	500	4725/2364/3936

Table 5.1: Summary statistics of three citation datasets and the size of train/val/test sets used in this work.

Experiments

We adopt and slightly modify the OCGNN setup in [120] to examine the performance of popular GNN frameworks, namely GCN [48], GAT [116], GraphSAGE [37], and SGC [121] for our proposed rh-OCGNN paradigm. Three experiments are conducted to investigate the drawback of OCGNN and demonstrate the excellent performance of rh-OCGNN.

In the first experiment, three OCGNN models (OC-GCN, OC-GAT, OC-GraphSAGE) are replicated as in [120]. The hyperparameter β that controls the size of the sphere is set as 0.1. We use Glorot uniform weight initialization [36] to initialize the weights of the network. All models are

trained for a maximum of 5000 epochs using the AdamW optimizer [61] with a learning rate of 0.001 and a weight decay coefficient of 0.0005. The training process is allowed to stop early based on validation loss and AUC score with a patience of 100 epochs. For each OCGNN model, we use two different network structures: a three layers of sizes 64-64-32 (on Cora and Citeseer datasets) and a two layers of sizes 128-64 (on Pubmed dataset). Each hidden layer is followed by the ReLU activation function and then a dropout layer of 0.5 rate is applied. Pooling aggregator is adopted for each GraphSAGE layer. For OC-GAT, the number of attention head is set as 8.

The second experiment is designed to examine the effect of the hyperparameter β on the loss function and the radius of OCGNN using OC-GAT, OC-GCN, and OC-SGC. An exponential grid of 30 values of β from e^{-10} to e^{-0} are evaluated. We adopt the setting of the first experiment with two exceptions: the number of maximum training epochs is reduced to 2000, and the criterion for early stopping is changed to validation accuracy rate. For OC-SGC, we set the number of hops $k = 2$ and the number of output units as 32. OC-GCN and OC-GAT network structures are kept the same as in the first experiment.

The last experiment illustrates a typical tuning process for the rh-OCGNN paradigm. Three rh-OCGNN models (rh-OC-GAT, rh-OC-GCN, and rh-OC-SGC) are evaluated. For hyperparameter r , an exponential grid of 30 values from e^{-2} to e^{-10} are used. We adopt the same setting as in the second experiment for common hyperparameters, except that the number of maximum training epochs is reduced to 1000. The network structures of rh-OC-GAT, rh-OC-GCN, and rh-OC-SGC are kept the same as in the second experiments.

We implement our proposed framework (rh-OCGNN) and OC-SGC using PyTorch [77] and Deep Graph Library [118]. Other OCGNN models implementation is adopted and edited from [120]

Github repository ¹. We release the source code for all models in our Github repository ² for reproducibility purposes. All experiments are carried out using Kaggle kernels ³ and Azure ⁴ Standard NC6 Compute Instance with 1 GPU NVIDIA Tesla K80.

Results

Table 5.2 shows the results of our first experiment where we replicate three OCGNN models (OC-GAT, OC-GCN, and OC-SAGE) that have been evaluated in [120]. The average AUCs are similar to the original results reported in the previous study. AUC assumes values from 0 to 1 and serves as an aggregate measure of the classifier’s capability to discriminate between normal objects and anomalies. In practice, AUC of 0.7 or less implies poor discrimination, 0.7 to 0.8 suggests acceptable discrimination, 0.8 or above indicates excellent discrimination [40]. OC-GAT is the most desirable GAD framework on Cora for its stellar discriminating power (AUC of 89.57%), while OC-SAGE appears to be a more effective framework on both Citesser and Pubmed (AUCs of 83.10% and 77.07% respectively). Our experimental exploration reaffirms the performance of the state-of-the-art OCGNN. Although it appears that the models achieve somewhat decent performance, one should be aware of the insufficiency of AUC as the sole measurement of accuracy for binary classification task [60]. AUC was designed to provide a summary measurement of a classifier’s predictive power across all possible choices of threshold. The intention was to prevent any subjectivity in threshold choice from influencing accuracy evaluation. However, for anomaly detection problem discussed in this work, the model has an intrinsic threshold, i.e. the radius of the description sphere, and hence, its performance should not be evaluated using AUC alone.

¹<https://github.com/WangXuhongCN/OCGNN>

²<https://github.com/phuongpho/rh-OCGNN>

³<https://www.kaggle.com/code>

⁴<https://azure.microsoft.com/en-us/>

Method	Cora	Citeseer	Pubmed
Our experiment			
OC-GAT	89.57 \pm 0.05	76.15 \pm 0.03	57.76 \pm 0.01
OC-GCN	79.74 \pm 0.07	64.11 \pm 0.03	53.69 \pm 0.02
OC-SAGE	86.81 \pm 0.06	83.10 \pm 0.13	77.07 \pm 0.01
Results in [120]			
OC-GAT	88.19 \pm 0.02	79.06 \pm 0.03	60.98 \pm 0.01
OC-GCN	73.25 \pm 0.02	62.81 \pm 0.01	54.53 \pm 0.01
OC-SAGE	86.97 \pm 0.04	85.62 \pm 0.01	74.72 \pm 0.03

Table 5.2: Results of the first experiment where we replicate OCGNN with three popular GNN frameworks - GCN, GAT, and GraphSAGE. Test AUCs (in percentage) averaged over 10 independent initializations and their corresponding standard deviations. Note that the standard deviations are not converted to percentage to keep consistent with [120]. The best results are highlighted in boldface. OC-SAGE outperforms its peers on Citeseer and Pubmed datasets while OC-GAT achieves highest average AUC on Cora dataset.

Figure 5.3 shows the performance of the OCGNN models in the first experiment on three other predictive metrics: accuracy, precision, and recall rates. Given the balanced nature of the test sets (described in Data section of this chapter), the low accuracy rates reveal the false impression of excellent performance left by the AUCs. We also notice an intriguing pattern where models achieve outstanding recall rates (except for OC-GAT on Pubmed) while average out mediocre precision rates (around 50%). Recall that these OCGNNs would classify an observation as outlier if it lies outside the hyper-sphere, i.e. its distance from the sphere center is greater than the radius. The above pattern implies that the fitted radii are too small, and the models effectively classify the majority of test data as outliers, since their distances easily exceed the radii. This result underscores the major drawback of the OCGNN, being its tendency to overfit the decision threshold, i.e. the sphere’s radius, in its attempt to minimize the training loss (Eq. 5.5).

Figures 5.4 and 5.5 show the results of the second experiment, in which we investigate the effect of $\beta \in (0, 1]$, an important hyperparameter of the OCGNN model, on the radius and classification capability. β regulates the proportion of training data ($\beta \times 100$ percent) to be mapped outside of the

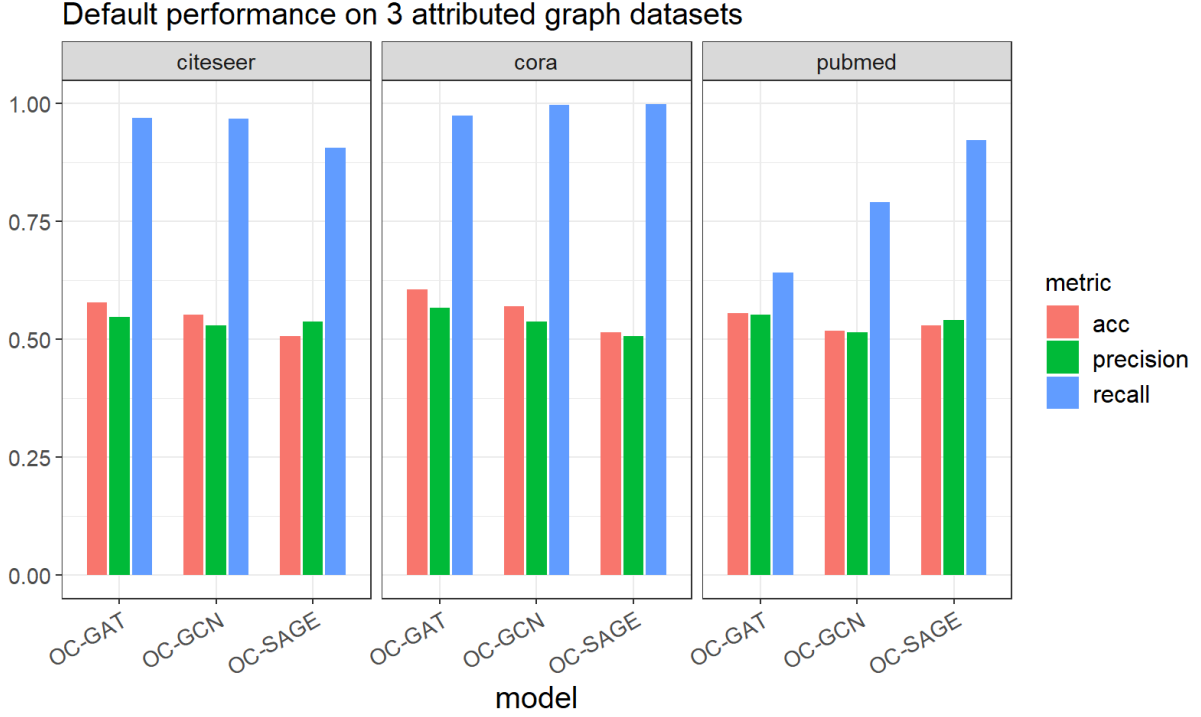


Figure 5.3: Average test accuracy, precision, and recall rates (over 10 independent initialization of OCGNN models on benchmark datasets. Albeit achieving promising AUCs (reported in Table 5.2, closer inspection reveals subpar performance of OCGNN models indicating sub-optimal decision threshold, i.e. radius, for anomaly detection task.

boundary and therefore controls the flexibility of the model by shrinking or expanding the radius of the sphere [120]. Eq. 5.31 reveals the role of β in determining r via percentage line search. Theoretically, an inverse proportional relationship between β and r should be expected, i.e. a large value of β results in small r and vice versa. However, figure 5.4 discloses a rather counter intuitive behavior of the radius as β varies over its domain. Overall, the radius exhibits downward trend with occasional upward deviation or sharp spikes. This tendency becomes noticeable at small values of β (from -10 to -2.5 on the logarithmic scale of β). Since tiny β dictates large values of r , the majority of training nodes are mapped inside the boundary. Hence, the loss function takes into account almost no training samples in the penalty component (the first term in Eq. 5.5). The

network might not be able to learn a good mapping to a latent space capable of clustering the normal samples compactly around the center. As a consequence, the radius learned on training distances produced on such space might behave sporadically. Furthermore, as different values of β result in distinct optimal embedding spaces learned by OCGNN, the magnitude of distances might be inconsistent between these latent spaces. Hence, applying percentage line search (Eq. 5.31) on these distance set $\mathbf{d}_{v_{tr}}$ will produce incompatible radii.

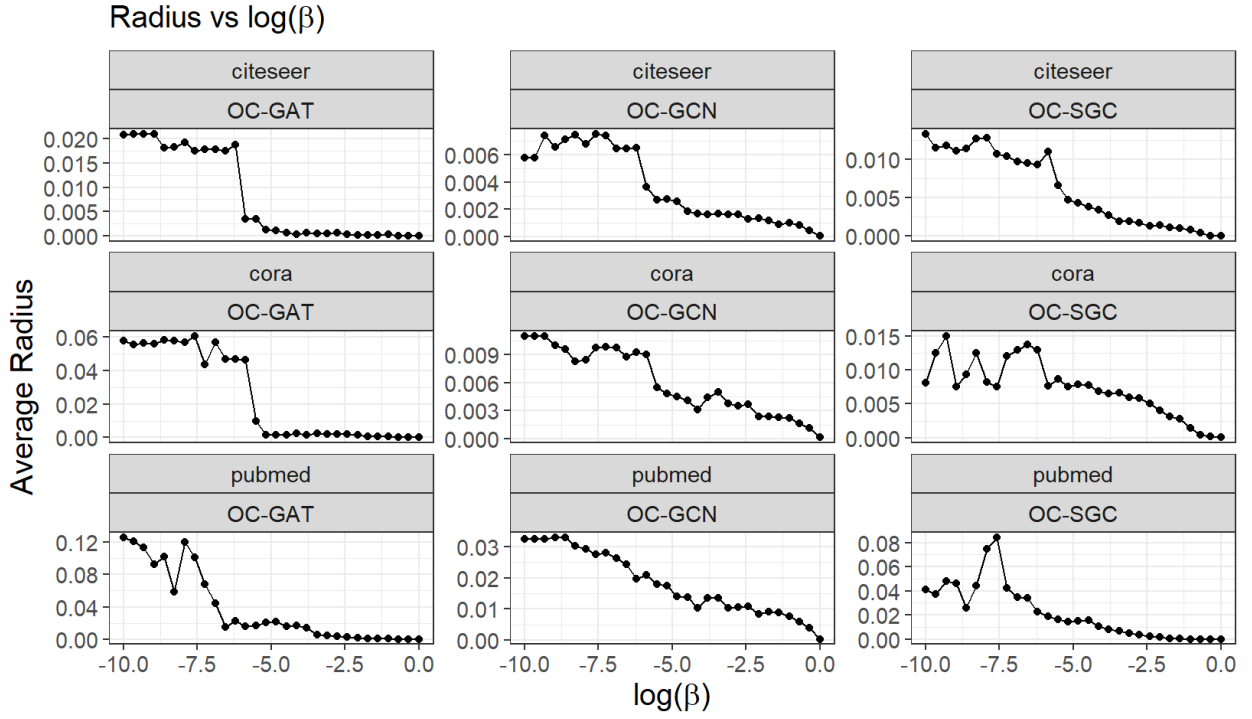


Figure 5.4: Average radius (over 5 independent initialization) against a range of β (on log scale). The radius is an intrinsic decision threshold to detect anomalous nodes in OCGNN framework. The plot shows sporadic behavior of radius indicating that the mapping learned by OCGNN is not efficient to represent normal class data.

Figure 5.5 examines the effect of β on the training loss. A recurrent pattern can be seen as the loss starts at high values and gradually decreases to the vicinity of zero. Notice that while the radius remains relatively stable, the training loss decreases rapidly at small values of β (from -10 to

-7.5 on the logarithmic scale of β). As β varies over the lower end of its spectrum, the weight of the penalty component decreases ($1/\beta$ in Eq. 5.5) but there might be no change in the number of training samples contributing to the loss. With little to no change in training samples, the model might learn similar mapping spaces which result in similar radii. A high value of the loss function indicates that the model is unsuccessful in learning a good representation of the normal class due to a restricted contribution of the small proportion of training sample. It confirms the observation above that setting β too small is ineffective for the training process.

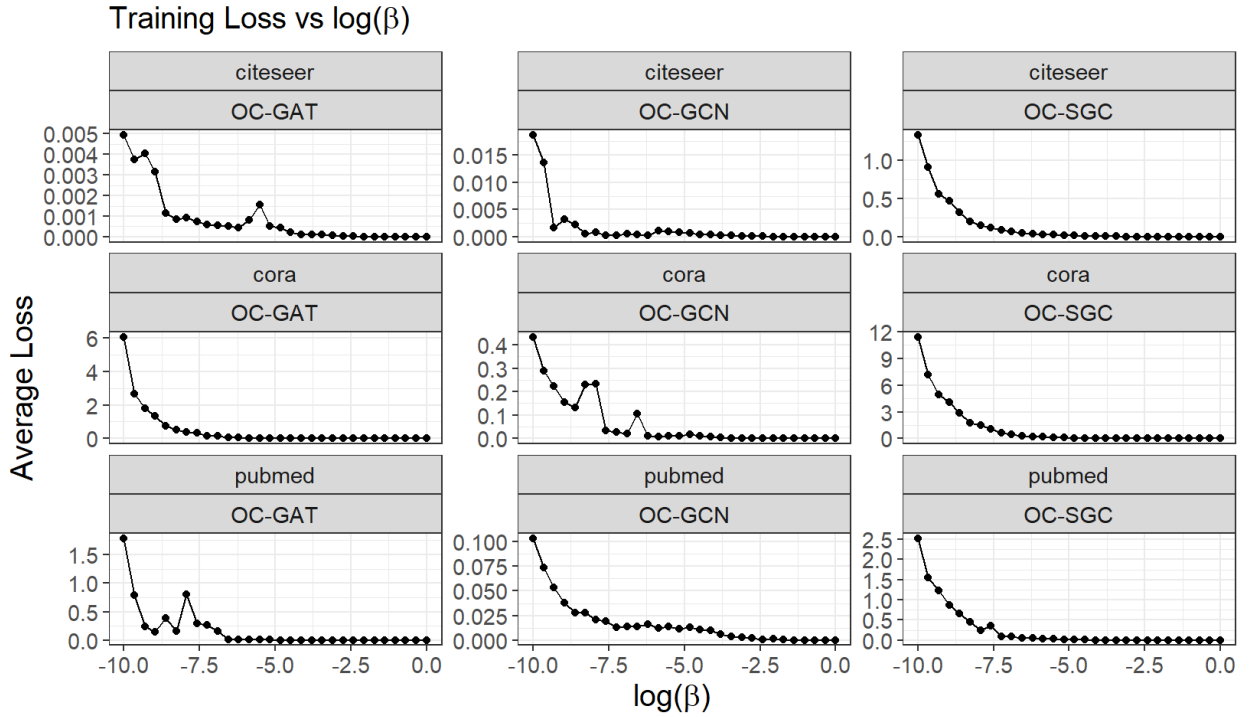


Figure 5.5: Average training loss curves (over 5 independent initialization) against a range of β (on log scale). Large losses occur at small values of β indicate the model is unsuccessful in learning representative embedding space for normal class data.

In contrast to OCGNN, our framework treats the sphere’s radius as a hyperparameter and removes its direct role in optimization of the training loss (Eq. 5.41). This modification gives the users more

leeway to exert domain specific control on the radius which also serves as the decision threshold of their anomaly detection system. As with many anomaly detection applications, one needs to tune the model hyperparameters to achieve the best performance [47, 57, 76, 127]. In rh-OCGNN, the radius is obviously one of the most important hyperparameters. For a specified radius, the model produces a latent space that can closely cluster all training samples around the center and within the boundary of the sphere.

Figure 5.6 illustrates the tuning process of three rh-OCGNN models and shows their behaviors with respect to various settings of radius (shown in logarithmic scale). Small values of radius facilitate the training process as more training samples are accounted that contribute to the penalty component in the loss function (the first term in Eq. 5.41). Once r becomes large, the amount of contributing training samples declines and the model fails to generate a good embedding space that compactly maps the normal samples. Hence, the training loss gets larger as r increases.

One might notice similarity in the behavior of rh-OCGNN (figure 5.6) and OCGNN (figure 5.5). In general, as the number of training samples mapped outside the sphere increases (by increasing β in OCGNN or decreasing r in rh-OCGNN), the model gets better at learning node embedding to minimize the loss consisting of penalized distances of these instances. This results in a good mapping space and a hypersphere capable of describing normal class data. However, it is worth to note that our framework rh-OCGNN exerts exact control over the sphere's volume by directly specifying its radius. On the other hand, OCGNN is only capable of affecting volume via β . Since β dictates the proportion of training samples mapped outside the sphere, different values of β might result in similar sphere's volumes (i.e. same radii) as long as the embedding spaces satisfy the specified fraction of training samples violating the boundary.

Figure 5.7 compares the tuning process of OCGNN and rh-OCGNN. The plots show validation accuracy curves with respect to various values of hyperparameter (β for OCGNN and r for rh-

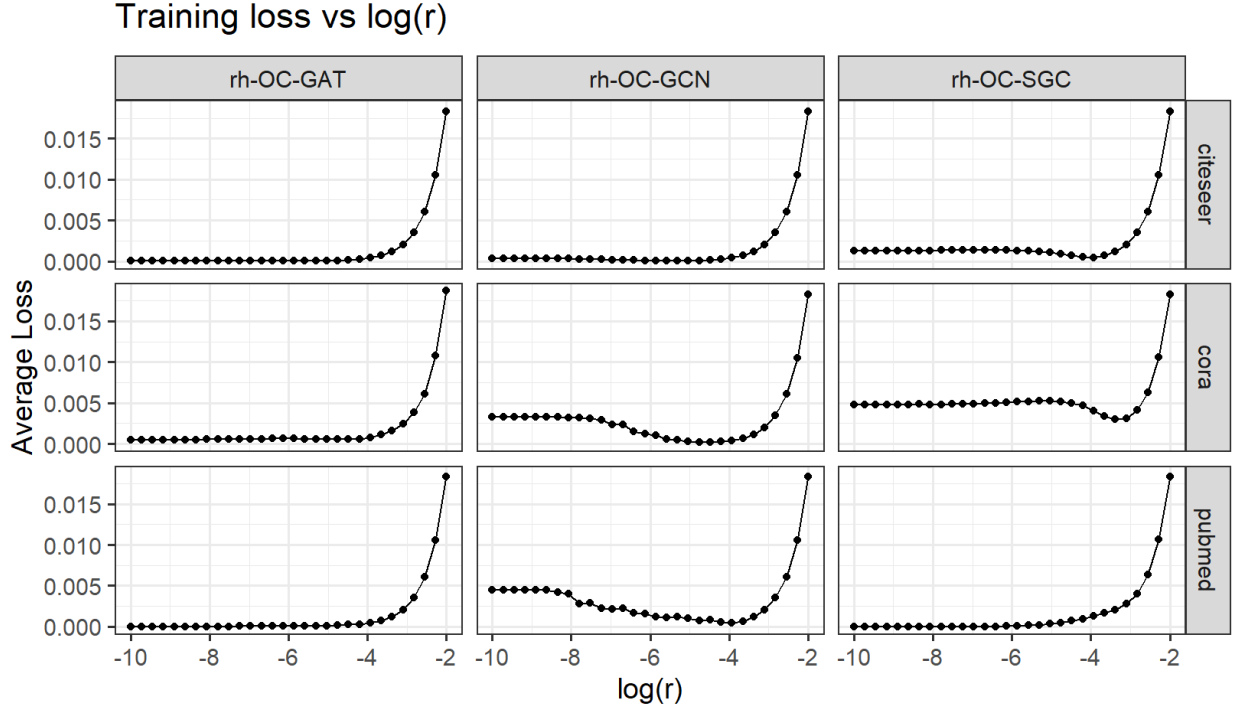


Figure 5.6: Average training loss curves (over five independent initialization) against a range of radius (on log scale). As the radius gets larger, less training samples are considered in the training process resulting to larger loss values.

OCGNN). Our rh-OCGNN framework provides great flexibility for practitioners to fine-tune the decision threshold, i.e. the radius, for their anomaly application. One can easily employ a typical validation approach to select the optimal r based on any choice of accuracy metrics on the validation set. As opposed to the jagged and multimodal validation curves produced by OCGNN, our validation curves appear to be unimodal which further facilitates tuning process as there is a unique optimal value available for consideration. For this experiment, we choose the optimal r at which the validation accuracy rate is maximized.

Table 5.3 compares the best performance of rh-OCGNN and OCGNN on test accuracy rates. Important hyperparameters of rh-OCGNN (r) and OCGNN (β) are tuned and optimized with respect

to validation accuracy rates. Unlike AUC, accuracy rate is a single-number measurement summarizing the proportion of correctly predicted samples with respect to a specific decision threshold. It is a de facto metric to evaluate model performance in classification problem and is especially desirable for balanced test data as in our experiments (described in 2). Our experiment shows that the rh-OCGNN models outperform their OCGNN counterparts on all benchmark datasets. rh-OC-SGC shines with its capability of correctly classifying 71.74% and 63.44% of normal and anomalous nodes on the Cora and Pubmed dataset, respectively. On the other hand, rh-OC-GAT makes 85.65% of accurate classification on Cora and hence, surpasses all of its peers. This result highlights the advantage of rh-OCGNN as it allows users to obtain an optimal decision threshold for the anomaly detection task and thus can achieve better performance.

Figure 5.8 further investigates the best performance of rh-OCGNN and OCGNN on two other accuracy metrics, precision and recall rates. Precision quantifies the number of samples predicted as event that are truly event while recall denotes the number of true events correctly classified as event. In anomaly detection problem, the event of interest is anomaly and a model with high precision and recall rates is desirable. However, one should be wary of wide disparity between precision and recall as it indicates a poor choice of decision threshold. For example, high recall and low precision rates suggest that the threshold is too low and the model effectively detects all test data as anomaly. We observe sharp contrast between recall and precision rates on OCGNN models, indicating failure to obtain optimal decision thresholds even after tuning the hyperparameter β . On the other hand, despite slightly lower recall rates, the harmony in recall and precision metrics observed on rh-OCGNN suggests that our framework achieves satisfactory threshold, i.e. radius, for anomaly detection tasks. As a result, rh-OCGNN obtains significantly higher accuracy rates on Cora and Citeseer datasets. Pubmed appears to be a greater challenge, as both frameworks have slightly above average performance. Due to its larger scale (highest number of nodes and connections, described in Table 5.1), there might be substructures existing within the normal class

Dataset	Model	beta	Radius	Test Accuracy rate
rh-OCGNN				
Citeseer	rh-OC-GAT	NA	0.00072	69.40 \pm 2.00
	rh-OC-GCN	NA	0.00164	62.51 \pm 2.50
	rh-OC-SGC	NA	0.00651	71.74 \pm 2.69
Cora	rh-OC-GAT	NA	0.00375	85.65 \pm 3.90
	rh-OC-GCN	NA	0.00375	69.00 \pm 5.23
	rh-OC-SGC	NA	0.01489	82.01 \pm 2.02
Pubmed	rh-OC-GAT	NA	0.00031	54.60 \pm 0.68
	rh-OC-GCN	NA	0.01489	55.13 \pm 0.91
	rh-OC-SGC	NA	0.00164	63.44 \pm 0.57
OCGNN				
Citeseer	OC-GAT	0.01596	0.00035	55.44 \pm 7.66
	OC-GCN	0.04489	0.00166	55.78 \pm 3.86
	OC-SGC	0.00402	0.00654	70.94 \pm 5.58
Cora	OC-GAT	0.01596	0.00260	65.65 \pm 10.07
	OC-GCN	0.01130	0.00411	63.54 \pm 5.66
	OC-SGC	0.00018	0.00933	69.09 \pm 5.13
Pubmed	OC-GAT	0.50175	0.00012	55.53 \pm 0.26
	OC-GCN	0.01596	0.01038	55.15 \pm 0.92
	OC-SGC	0.04489	0.00492	62.71 \pm 0.52

Table 5.3: Test accuracy rates (%) averaged over five independent initialization of rh-OCGNN and OCGNN. The optimal values of β (for OCGNN) and r (for rh-OCGNN) are selected via tuning to maximize the validation accuracy rates. Note that on OCGNN, the radius is not a hyperparameter and is learned during training phase. The best models are highlighted with boldface. Overall, the rh-OCGNN models outperform their OCGNN peers on all benchmark datasets. rh-OC-SGC shines on Cora and Citeseer while rh-OC-GAT achieves best performance on Cora.

that cannot be well described by one single hypersphere. Although both frameworks struggle on Pubmed dataset, rh-OC-SGC achieves better threshold resulting to higher accuracy as well as greater balance between precision and recall rates.

Conclusion

In this paper, we closely examine the performance of the existing OCGNN methodology for anomaly detection on graph structured data and propose an alternative framework called the "radius as hyperparameter OCGNN" (rh-OCGNN). Our model builds a description of normal class data by constructing a hypersphere with a given radius to enclose all training samples on the GNN latent node embedding space. The advantage is twofold: not only does it allow for the incorporation of domain knowledge in establishing decision threshold for anomaly detection task (in the form of the sphere's radius), users can easily optimize their applications via fine-tuning the threshold with respect to any performance metric of their desire. Three experiments are conducted: the first one serves as a striking illustration of the pitfall of utilizing inadequate accuracy metric (AUC) for model benchmarking, especially for the class of models with some intrinsic threshold. The second experiment demonstrates the weakness of the state-of-the-art framework, while the third experiment shows the improved performance of the proposed model.

The newly proposed model provides improved accuracy and precision on Cora and Citeseer datasets. It also shows a validation accuracy curve for the key hyperparameter following a unimodal form which is expected for such a problem. This framework can also assist in monitoring event streams for changes of interest where input graphs are produced over a time-ordered stream based on an entropic measure for the time window [64]. Future work entails learning multiple hyperspheres for large datasets where groups sharing the same label can have multiple centers such as in fraud detection data.

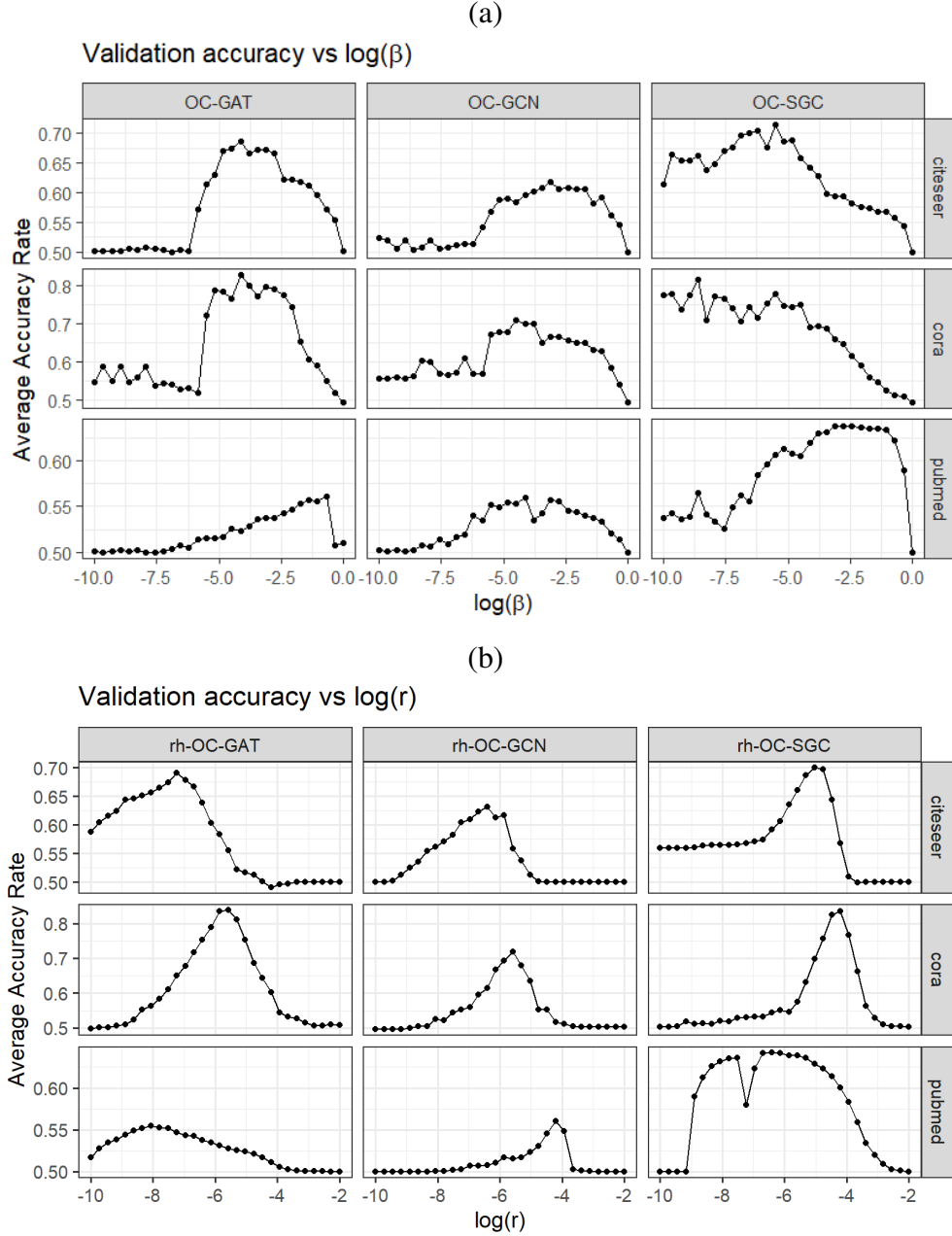


Figure 5.7: Tuning process of OCGNN (sub-figure A) and rh-OCGNN (sub-figure B). The plots show average validation accuracy rate curves (over five independent initialization) against a range of hyperparameter (β for OCGNN in sub-figure (a) and r for rh-OCGNN in sub-figure (b)). The proposed rh-OCGNN produces relatively smooth mount-shape validation curves facilitating the tuning process. The optimal radii can be easily selected where the curves reach their peaks.

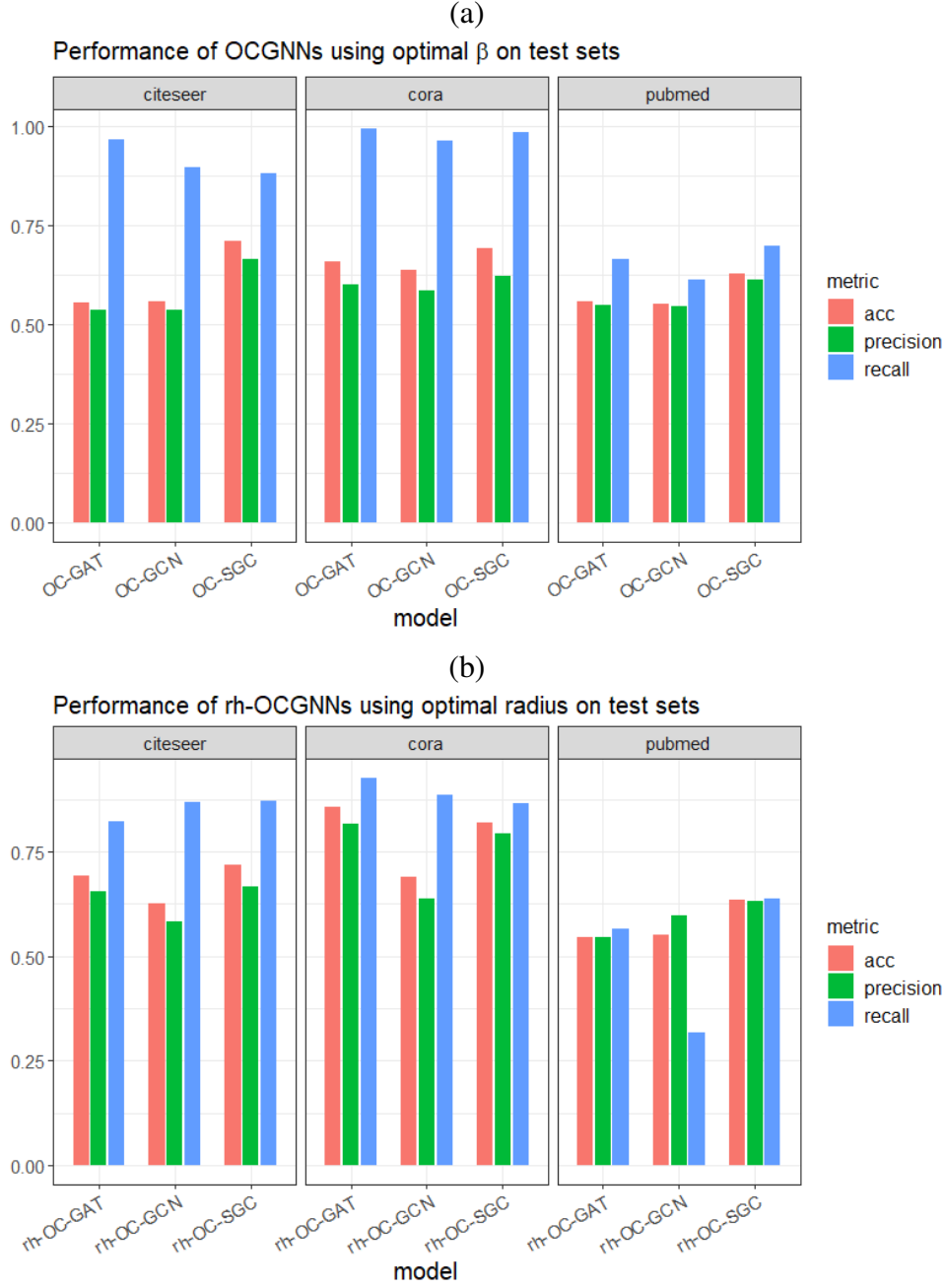


Figure 5.8: Average test accuracy, precision, and recall rates (over 5 independent initialization) of OCGNN (sub-figure A) and rh-OCGNN (sub-figure B). Important hyperparameters of OCGNN (β) and rh-OCGNN (r) are tuned using validation approach where the optimal values are selected to maximize validation accuracy rates. rh-OCGNN renders flexibility to directly tailor the decision threshold, i.e. the radius. Hence, it achieves better performance on all benchmark datasets.

CHAPTER 6: LINK PREDICTION WITH SIMPLE GRAPH CONVOLUTION AND REGULARIZED SIMPLE GRAPH CONVOLUTION

Introduction

Attributed graphs (networks) provide powerful representations of real-life complex systems where each element is regarded as node a with associated attributes (features) and the connectivity information between elements forms edges. These graph-structured feature data are used to represent complex systems in various domains such as social science (social networks [29]), biology (biochemical pathways [17]), material science (molecular networks [34]).

Link prediction aims to infer unknown connectivity in the form of a link between a pair of nodes in the graph. It has a wide range of applications such as friend recommendation in social networks [26, 1], knowledge graph construction [86], and protein-protein interaction [65].

Simple approaches to link prediction rely on heuristics to assess the similarity in network topology between a pair of nodes and determine the likelihood of link existence. The common-neighbor method uses the number of common neighbors to predict the link between a pair of nodes while preferential attachment [9] predicts that popular nodes are more likely to connect. The Adamic-Adar algorithm also utilizes the common neighbors and their popularity to predict connectivity between target nodes [1]. PageRank employs the idea of random walk model to score the link between the target nodes by taking into account all walks starting from one end to the other [75]. These heuristics can only capture limited network topologies and also fail to account for supplemental information in the form of node attributes.

Graph Neural Networks (GNNs) are neural networks operating on the graph domain and is able to combine node attributes and connectivity to produce informative node representations (node embeddings). GNNs produce expressive node representations (node embeddings) via an iterative message passing mechanism where each node aggregates its direct neighbors' feature vectors and updates its own feature vector with the aggregate information. The final node representation encapsulates structural information of a k -hop neighborhood. To produce a representation of the entire graph, one simply applies a pooling operator (such as summation) on the set of feature vectors of all nodes [34, 126]. Many GNN variants have been proposed and have achieved state-of-the-art performance in a variety of tasks [123]. Graph Convolutional Networks (GCN) [48] learn a graph representation via layer-wise propagation rules representing localized spectral filters. Graph Attention Networks (GAT) [116] utilizes an attention mechanism to account for node neighbors' importance in the aggregation phase. The Simple Graph Convolution (SGC) [121] simplifies the GCN by removing non-linear transitions between layers while retaining its representational power. The work of [79] demonstrates the expressive node embedding capability of SGC and explores flexible regularization mechanisms to facilitate meaningful interpretation.

Researchers have studied GNN applications for link prediction task. The work of [129] extracts a local subgraph around each target link and utilizes GNN to learn a function mapping the subgraph patterns to link existence. [4] propose a novel Variational Graph Normalized AutoEncoder (VGNAE) which aims to improve the embedding of isolated nodes by using L2-normalization.

In this work, we extend SGC and regularized SGC for a link prediction task. Since the SGC framework involves mostly linear operations, it offers high efficiency and scalability compared with other frameworks relying on more layers in order to improve accuracy. It is also of interest to investigate the effect of regularization on dimensionality reduction to highlight important edge embeddings and improve our understanding of these embeddings produced by SGC on the assessment of connectivity between target nodes. Our framework can be used to build a recommendation

system for large social networks where computational cost is high, and thus a fast and efficient framework is preferable. Our model can highlight useful edge embeddings that are important for determining connectivity between users.

Methodology

SGC

SGC is introduced in [121] as a simplified GNN model developed from GCN [48] by removing non-linear activation functions between hidden layers and reparametrizing successive layers into one single layer. This modification reduces superfluous complexity of the GCN while retaining its superb performance on many downstream tasks. In this section, we briefly present the original SGC.

An attributed graph data set contains a graph $G = (V; E)$ of N nodes (vertices) $V = (v_1, v_2, \dots, v_N)$ and edges $(v_i, v_j) \in E$. Often, the connectivity in E is represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ where each element a_{ij} represents an edge between node v_i and v_j ($a_{ij} = 0$ if v_i and v_j are disconnected). Each node might have a feature vector $x_{v_i} \in \mathbb{R}^D$ which is stacked together to form a node attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$. We define the degree matrix $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_N)$ as a diagonal matrix whose off-diagonal elements are zero, and each diagonal element d_i captures the degree of node v_i and $d_i = \sum_j a_{ij}$. Each node i receives a label from the C classes and therefore can be coded as one hot vector $y_i \in \{0, 1\}^C$.

The GCNs and SGC add self-loops and normalize the adjacency matrix to get the matrix \mathbf{S} :

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \quad (6.1)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \text{diag}(\tilde{\mathbf{A}})$. This normalization allows successive powers of the matrix to not influence the overall size the projections. The SGC removes non-linear transformation from the k^{th} layer of the GCN, resulting in a linear model of the form:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S} \dots \mathbf{S} \mathbf{X} \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}). \quad (6.2)$$

The SGC classifier is then achieved by collapsing the repetitive multiplication of matrix \mathbf{S} into the k^{th} power matrix \mathbf{S}^K and reparameterizing the successive weight matrices as $\Theta = \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)}$:

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{S}^K \mathbf{X} \Theta). \quad (6.3)$$

The parameter k corresponds to the number of 'hops' which is the number of edge traversals in the network adjacency matrix \mathbf{S} . This parameter k can be thought of as accumulating information from a certain number of hops away from a node (as visually described in [121]). If $k = 0$ the methodology becomes equivalent to a logistic regression application that is known to be scalable to large datasets. Since the SGC introduces the matrix \mathbf{S} in a linear operation, the same scalability applies. The weight matrix Θ is trained by minimizing the loss:

$$\mathcal{L} = \sum_{l \in \mathcal{Y}_L} \sum_{c \in C} Y_{lc} \ln \hat{Y}_{lc} \quad (6.4)$$

where \mathcal{Y}_L is a collection of labeled nodes.

Regularized SGC

The work of [79] illustrates SGC's expressive power in the node classification task and proposes a flexible regularization methodology to reduce the number of parameters and highlight a sparse set of important features. They introduce a flexible set of constraints in terms of shrinkage parameters

L_1, L_2 , and L_3 in the loss \mathcal{L} from Eq 6.4:

$$\begin{aligned} \mathcal{L}_R = \mathcal{L} + L_1 \times \sum_{c \in C} \left(\sum_{d=1}^D |\Theta_{R(\cdot, c)}|^4 \right)^{(-1)} + L_2 \times \sum_{c \in C} \|\Theta_{R(\cdot, c)}\|_2 + \\ L_3 \times \left(\sum_{c_1 \in C} \sum_{c_2 \in C} \left(|\Theta_{R(\cdot, c_1)}^T| \cdot |\Theta_{R(\cdot, c_2)}| : c_1 \prec c_2 \right) \right) \end{aligned} \quad (6.5)$$

where Θ_R is the parameters for the regularized fitted SGC and $|\Theta_{R(\cdot, c)}|^4$ denotes the normalized vector for each class projection in the parameter matrix (which are columns) and that each element is raised to the power of 4. L_1 regulates the number of parameters by inducing penalization with a larger skew in the number of elements in the columns of Θ_R . The L_2 term controls the total magnitude of the parameter vector, while the L_3 term penalizes the class label projections, which have large overlaps.

Link prediction

Given a graph $G_O = (V; E_O)$, the set of observed links (edges) E_O can be considered as a subset of the unobserved true links E^* . Link prediction involves distinguishing between the true and false links from a set of candidate links $E^C = \{(v_i, v_j)\}$ where v_i and v_j are unconnected nodes in V .

A link predictor $LP = f : V \times V \rightarrow \mathbb{R}$ is a mapping function to classify the candidate link from a given pair of nodes. A simple mapping function utilizes heuristic node similarity such as common neighbors, Adamic-Adar [1], etc. to access the similarity between a pair of nodes and produces a score as a link likelihood, which can be used to classify the given link as true or false. The drawbacks of these heuristics are 1) they are handcrafted features that only capture limited network topologies and 2) they ignore rich information from node attributes.

GNNs are powerful neural network frameworks capable of combining node attributes and con-

nectivity to solve various downstream tasks such as node classification, link prediction, and graph classification [136]. The success of the GNNs comes from employing a message passing mechanism which iteratively aggregates neighborhood representations to learn an expressive node embedding [126, 34]. We denote a L -layer GNN model as $g(\mathbf{X}, \mathbf{A}; \Omega)$ where the input consists of a nodes attribute matrix \mathbf{X} and an adjacency matrix \mathbf{A} . $\Omega = \{\Omega^{(1)}, \Omega^{(2)}, \dots, \Omega^{(L)}\}$ is the set of network weights. The model outputs node embedding $\mathbf{Z} \in \mathbb{R}^{N \times F}$ where each row $z_{v_i} \in \mathbb{R}^F$ is vector representation of node v_i .

In this work, we utilize two GNN frameworks, namely SGC [121] and regularized SGC [79] to learn an informative node representation $\mathbf{Z} = \mathbf{S}^K \mathbf{X} \Theta$. For a candidate edge $e_{ij}^C = (v_i, v_j)$, its edge embedding $z_{e_{ij}} \in \mathbb{R}^F$ is produced by combining node representations z_{v_i} and z_{v_j} using the Hadamard product, i.e. $z_{e_{ij}} = z_{v_i} \circ z_{v_j}$. A link predictor in the form of a one-layer fully connected neural network $f(z_{e_{ij}}, W) = \sigma(z_{e_{ij}} \times W)$ can be optimized to learn a good mapping function to classify the candidate edges as true or false. This choice of link predictor makes it effectively a logistic regression model where the effect of the edge embeddings can be easily interpreted via the set of weights W .

Given a training set E^{tr} including true (positive) and false (negative) link samples, the model attempts to minimize the loss function:

$$\mathcal{L}(\Theta, W) = - \sum_{e \in E^{tr}} y_e \log(\hat{y}_e) + (1 - y_e) \log(1 - \hat{y}_e) \quad (6.6)$$

Motivated by the regularized SGC [79], we adopt the penalty terms into the loss:

$$\mathcal{L}_R(\Theta, W_R) = \mathcal{L} + L_1 \times \left(\vec{1} \cdot \hat{W}_R^4 \right)^{(-1)} + L_2 \times \|W_R\|_2 \quad (6.7)$$

where \hat{W}_R denotes normalized weight vector of the link predictor.

This modification would produce a sparser set of weights W_R , highlighting the important edge embeddings for link prediction. Another benefit of regularization is to improve model performance by reducing overfitting.

Datasets

We employ three popular and publicly available attributed graph datasets namely Cora [67], Citeseer [33], and Pubmed [94] for the link prediction task using SGC and regularized SGC. These datasets contains scientific publications (regarded as nodes) where each paper is represented by a binary vector indicating the presence of informative keywords (regarded as node’s features) and the citations between papers form edges. Each paper can be categorized into a specific topic (regarded as label). Datasets statistics are given in Table 6.1.

Since the link prediction problem can be considered as a binary classification of edges into positive and negative classes, we randomly select existing edges to form positive samples. Negative samples are then chosen by randomly sampling pairs of unconnected nodes. The amount of samples between two classes is kept balanced. Each set of samples is partitioned and combined to form the train, validation, and test sets with the proportions of 70/10/20%, respectively.

Dataset	Nodes/Edges/Classes	Node Features	Train/Val/Test
Cora	2708/5028/7	1433	3520/503/1055
Citeseer	3327/4614/6	3703	3230/461/935
Pubmed	19717/44327/3	500	31029/4433/8865

Table 6.1: Summary statistics of three citation datasets and the size of train/val/test sets used in this work.

Experiment

Two GNN frameworks, namely SGC [121] and regularized SGC [79] are adopted for our experiment with the link prediction task. For both models, we set the number of hops $k = 2$ and the number of output units as 256. All models are trained for 100 epochs using the AdamW optimizer [61] with a learning rate of 0.01. Optimal regularization parameters L_1 and L_2 are chosen from a grid search of 50×50 values, each varying from 10^{-5} to 10^2 , to maximize validation AUC.

Results

In this section, we discuss the application of the proposed methodology to the three citation datasets. The purpose is to examine the capability of both the SGC and the regularized SGC to the link prediction task to real-life data with a large number of features.

Figure 6.1 displays the weight vectors fitted on the Cora dataset using SGC (Subfigure (a)) and regularized SGC (Subfigure (b)). Each plot contains a histogram showing the weight vector’s distribution and a heat map illustrating its magnitude. In Subfigure (a), the fitted weight vector W ranges from -0.75 to 1.25 with a mode of 0.8 . The model utilizes all edges’ representations to produce discriminative scores for link classification and achieves a good AUC of 0.768 . Most weight values are centered around 1 , indicating that, under the logistic model, most edge embeddings have an equivalent impact on the link likelihood. Analogously, the same set of results produced with the regularized SGC is shown in Subfigure (b). The components of the weight vector W_R are slightly reduced and the model achieves a moderate AUC of 0.776 . It is clear that the penalty term L_2 on the total magnitude of the weight vector can only produce a weak shrinkage effect. Moreover, the regularization appears to limit overfitting and improve model performance.

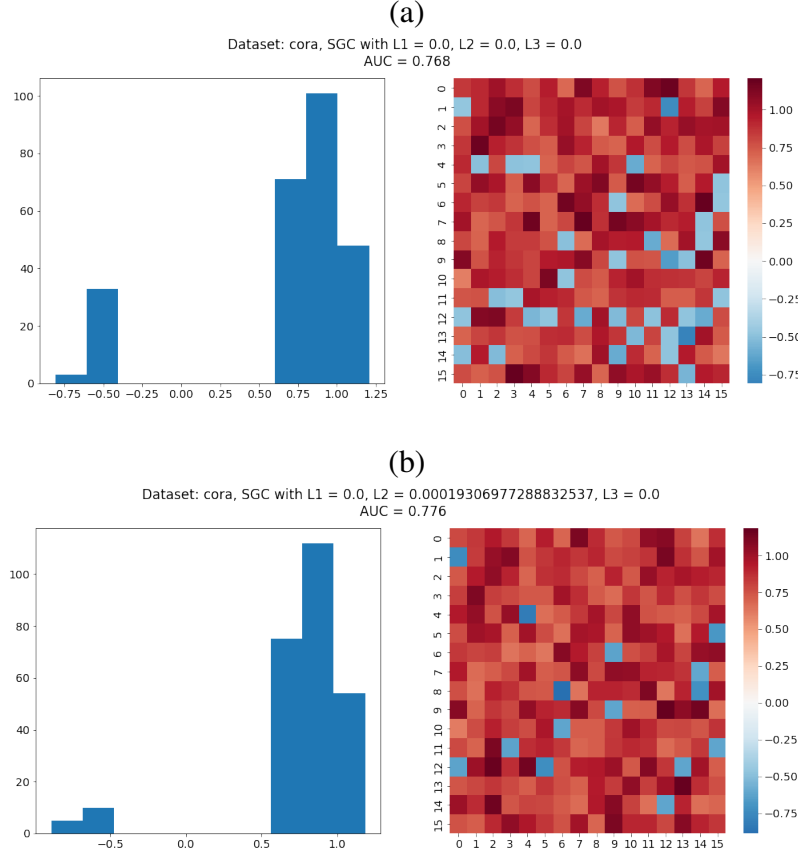


Figure 6.1: The plots show the results of applying SGC with and without regularization on the hidden edge embeddings of the Cora dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector’s magnitude over the hidden edge embedding highlighting key variables.

Figure 6.2 shows the results of SGC (Subfigure (a)) and regularized SGC (Subfigure (b)) on the Citeseer dataset. The fitted weight of SGC shown in Subfigure (a) seems to follow a bimodal distribution with values varies from -1 to 1 . The information of all edge hidden features are critical for link prediction and the model is capable of predicting link existence with an AUC of 0.709. The weight distribution suggests that two thirds of edge embeddings have a positive impact on link likelihood, while the remaining embeddings tend to decrease link likelihood. Subfigure

(b) shows the results of regularization incorporated in SGC framework. We observe a moderate shift in the weight's distribution and the magnitude of weights seems to decrease slightly. Again, regularization renders a boost in model performance with an increase of 2% in AUC.

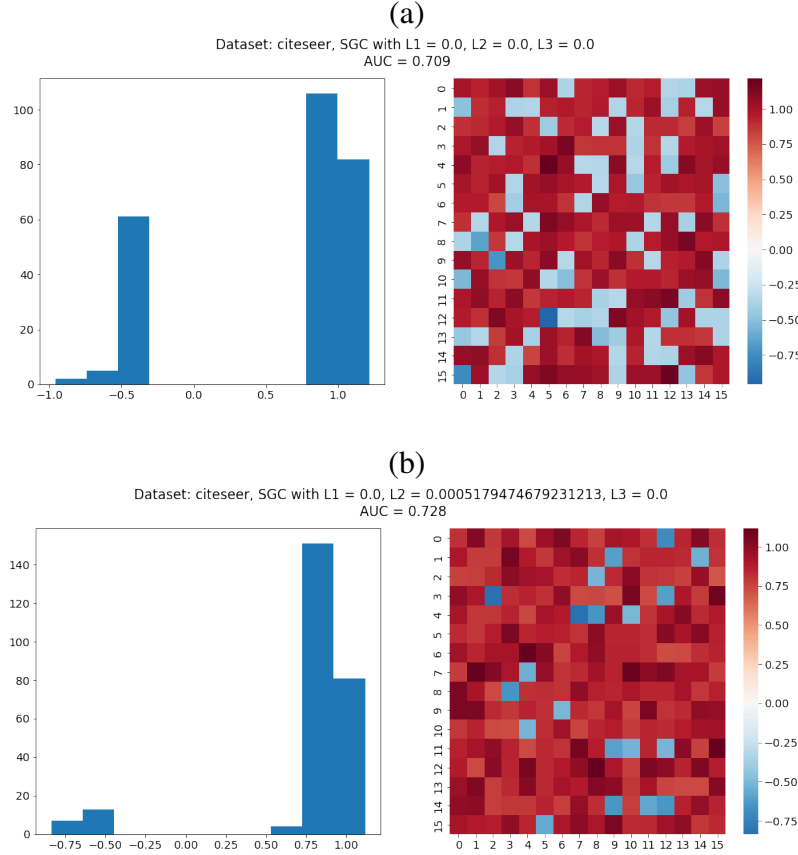


Figure 6.2: The plots show the results of applying SGC with and without regularization on the hidden edge embeddings of the Citeseer dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector's magnitude over the hidden edge embedding highlighting key variables.

Figure 6.3 displays similar results of SGC and regularized SGC on the Pubmed dataset. In the case of SGC, the model takes into account all edge's hidden features to produce effective scores to classify the edges. This results in a set of nonzero fitted weights shown in Subfigure (a) and

an excellent AUC of 0.913. Under the penalization of L_2 , the magnitude of the weight vector is reduced moderately as illustrated in Subfigure (b). We observe a shift from a fat-tailed distribution to a more centered distribution of weight values. Although a sparser set of weights is not achieved, our model still benefits from regularization as the AUC increases to 0.922.

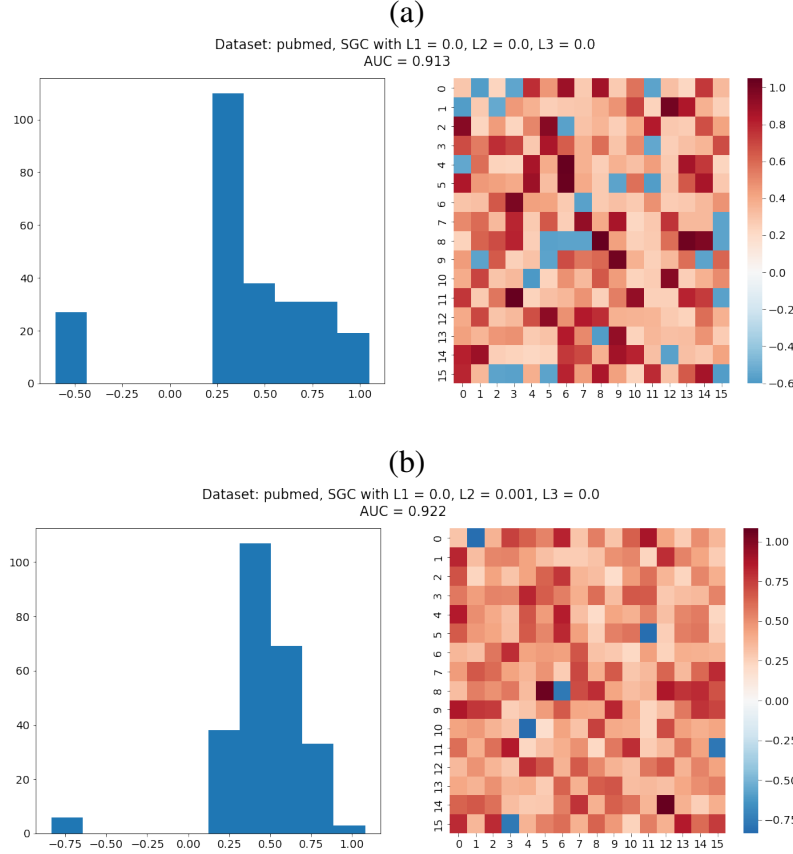


Figure 6.3: The plots show the results of applying SGC with and without regularization on the the hidden edge embeddings of the Pubmed dataset. The histogram in the left panel shows the distribution of weight and the heat map in the right panel displays the weight magnitude. Subfigure (a) presents the weight \mathbf{W} under the SGC model without regularization while Subfigure (b) shows the weight \mathbf{W}_R under the SGC model with regularization. It can be seen that the regularization reduces the weight vector’s magnitude over the hidden edge embedding highlighting key variables.

There are two possible reasons impeding model performance on Cora and Citeseer. These two datasets contain a higher number of classes than Pubmed. Node connectivity patterns may vary

significantly between each class. Utilizing a one-layer fully connected neural network as link predictor might fail to model a potential nonlinear relationship of node connectivity exists in these datasets. The simplicity of SGC might also be insufficient to produce highly expressive embeddings that are informative for distinguishing true and false links.

Conclusion

In this paper, we examine the capability of SGC and its regularized extension in link prediction on three citation attributed graph datasets. Although SGC renders a fast and efficient framework for computation, the model only achieves fair performance. This observation indicates the need of a nonlinear framework to successfully capture complex relationships in nodes' connectivity. The regularization proves to be an effective medium to reduce overfitting and helps improve model performance. Future work entails incorporating a flexible regularization scheme into a non-linear framework to improve both the performance and interpretability of the result.

CHAPTER 7: CONCLUSION

The main contributions of this dissertation are novel techniques and strategies to improve interpretability and efficiency of graph neural networks for node classification, link prediction, and anomalous nodes detection.

First, we have presented an extension of the SGC model allowing for a more explainable set of results to interested users in chapter 2 and in [79]. The regularization terms reduce the number of nonzero parameters, the overall magnitude of the parameter vectors, and the overlap between parameter vectors of the different classes. This extended framework inherits the fast and efficient properties of SGC yet renders a sparse set of parameters highlighting important node features defining class characteristics.

We then investigated optimal procedure to obtain training samples in order to improve the accuracy of node classification task on attributed graph data in chapter 3 and in [39]. Two sampling schemes were experimented in conjunction with three different measures of node importance. We explain the effect of these sampling strategies using the skewness of homogeneous connectivity distribution. We then proposed and empirically validated a heuristic measure to determine the best sampling strategy for the node classification task.

In the chapter 4 and in [38], we designed an comprehensive set of synthetic attributed graph datasets covering different network topology and node features and then evaluated the performance of the heuristic measure proposed above. We presented a quantitative guideline of how to use the heuristic measure to select best sampling strategy.

Next, we studied an existing work of one class GNN for anomalous node detection and proposed an improved model called the "radius as hyperparameter OCGNN" (rh-OCGNN) in chapter 5. Our

model allows users to apply domain knowledge in establishing decision threshold for anomaly detection task (in the form of the sphere's radius). Users can further optimize their applications via fine tuning the threshold.

Finally, we experimented the flexible regularization scheme of regularized SGC for link prediction task in chapter 6. The incorporation of regularization appears to reduce overfitting and improve model performance. However, the shrinkage effect was insignificant and the fitted models only achieved slightly above average performance. This observation suggests the need of a nonlinear framework to successfully capture complex relationship in nodes' connectivity.

LIST OF REFERENCES

- [1] Lada A Adamic and Eytan Adar. “Friends and neighbors on the web”. In: *Social networks* 25.3 (2003), pp. 211–230.
- [2] Charu Aggarwal, Gewen He, and Peixiang Zhao. “Edge classification in networks”. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE. 2016, pp. 1038–1049.
- [3] Nurzaman Ahmed, Hafizur Rahman, and Md I Hussain. “A comparison of 802.11 ah and 802.15. 4 for IoT”. In: *Ict Express* 2.3 (2016), pp. 100–102.
- [4] Seong Jin Ahn and MyoungHo Kim. “Variational Graph Normalized AutoEncoders”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2827–2831.
- [5] Leman Akoglu, Hanghang Tong, and Danai Koutra. “Graph based anomaly detection and description: a survey”. In: *Data mining and knowledge discovery* 29.3 (2015), pp. 626–688.
- [6] Réka Albert and Albert-László Barabási. “Statistical mechanics of complex networks”. In: *Reviews of modern physics* 74.1 (2002), p. 47.
- [7] Tim Althoff, Pranav Jindal, and Jure Leskovec. “Online actions with offline impact: How online social networks influence online and offline user behavior”. In: *Proceedings of the tenth ACM international conference on web search and data mining*. 2017, pp. 537–546.
- [8] Ana Isabel Rojão Lourenço Azevedo and Manuel Filipe Santos. “KDD, SEMMA and CRISP-DM: a parallel overview”. In: *IADS-DM* (2008).
- [9] Albert-László Barabási. “Network science”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987 (2013), p. 20120375.

- [10] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.
- [11] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.
- [12] Alexander Belyi et al. “Global multi-layer network of human mobility”. In: *International Journal of Geographical Information Science* 31.7 (2017), pp. 1381–1402.
- [13] Neda H Bidoki, Alexander V Mantzaris, and Gita Sukthankar. “Exploiting Weak Ties in Incomplete Network Datasets Using Simplified Graph Convolutional Neural Networks”. In: *Machine Learning and Knowledge Extraction* 2.2 (2020), pp. 125–146.
- [14] PV Bindu and P Santhi Thilagam. “Mining social networks for anomalies: Methods and challenges”. In: *Journal of Network and Computer Applications* 68 (2016), pp. 213–229.
- [15] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.
- [16] Stephen P Borgatti. “Centrality and network flow”. In: *Social networks* 27.1 (2005), pp. 55–71.
- [17] Pasquale Bove et al. “Prediction of Dynamical Properties of Biochemical Pathways with Graph Neural Networks.” In: *BIOINFORMATICS*. 2020, pp. 32–43.
- [18] Deborah Wright Brown and Alison M Konrad. “Granovetter was right: The importance of weak ties to a contemporary job search”. In: *Group & Organization Management* 26.4 (2001), pp. 434–462.
- [19] Enrico G Caldarola and Antonio M Rinaldi. “Big Data Visualization Tools: A Survey”. In: *Research Gate* (2017).
- [20] LF Carvalho et al. “A simple and effective method for anomaly detection in healthcare”. In: *Proceedings of the SIAM International Conference on Data Mining Workshop*. Vol. 2015. 2015, pp. 16–24.

- [21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.
- [22] L Chanussot et al. “The open catalyst 2020 (OC20) dataset and community challenges. arXiv”. In: *arXiv* (2010).
- [23] Paolo Crucitti, Vito Latora, and Sergio Porta. “Centrality measures in spatial networks of urban streets”. In: *Physical Review E* 73.3 (2006), p. 036125.
- [24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*. 2016, pp. 3844–3852.
- [25] Kaize Ding et al. “Deep anomaly detection on attributed networks”. In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 594–602.
- [26] Yuxiao Dong et al. “Link prediction and recommendation across heterogeneous social networks”. In: *2012 IEEE 12th International conference on data mining*. IEEE. 2012, pp. 181–190.
- [27] Ernesto Estrada. *The structure of complex networks: theory and applications*. Oxford University Press, 2012.
- [28] Leonhard Euler. “Solutio problematis ad geometriam situs pertinentis”. In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.
- [29] Wenqi Fan et al. “Graph neural networks for social recommendation”. In: *The World Wide Web Conference*. 2019, pp. 417–426.
- [30] Victor Fung et al. “Benchmarking graph neural networks for materials chemistry”. In: *npj Computational Materials* 7.1 (2021), pp. 1–8.

- [31] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep bayesian active learning with image data”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1183–1192.
- [32] Jing Gao et al. “On community outliers and their efficient detection in information networks”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 813–822.
- [33] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. “CiteSeer: An automatic citation indexing system”. In: *Proceedings of the third ACM conference on Digital libraries*. 1998, pp. 89–98.
- [34] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1263–1272.
- [35] Michelle Girvan and Mark EJ Newman. “Community structure in social and biological networks”. In: *Proceedings of the national academy of sciences* 99.12 (2002), pp. 7821–7826.
- [36] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feed-forward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.
- [37] William L Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *arXiv preprint arXiv:1706.02216* (2017).
- [38] Michael Hopwood, Phuong Pho, and Alexander V Mantzaris. “Exploring a link between network topology and active learning”. In: *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE. 2021, pp. 81–86.

- [39] Michael Hopwood, Phuong Pho, and Alexander V Mantzaris. “Exploring the Value of Nodes with Multicommunity Membership for Classification with Graph Convolutional Neural Networks”. In: *Information* 12.4 (2021), p. 170.
- [40] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.
- [41] Matthew O Jackson. “Networks in the understanding of economic behaviors”. In: *Journal of Economic Perspectives* 28.4 (2014), pp. 3–22.
- [42] Hawoong Jeong, Zoltan Néda, and Albert-László Barabási. “Measuring preferential attachment in evolving networks”. In: *EPL (Europhysics Letters)* 61.4 (2003), p. 567.
- [43] Dorina Kabakchieva. “Student performance prediction by using data mining classification algorithms”. In: *International journal of computer science and management research* 1.4 (2012), pp. 686–690.
- [44] Joseph Kahne and Benjamin Bowyer. “The political significance of social media activity and social networks”. In: *Political Communication* 35.3 (2018), pp. 470–493.
- [45] Nikhil Ketkar. “Introduction to pytorch”. In: *Deep learning with python*. Springer, 2017, pp. 195–208.
- [46] Mohamad Khedmati, Masoud Erfani, and Mohammad GhasemiGol. “Applying support vector data description for fraud detection”. In: *arXiv preprint arXiv:2006.00618* (2020).
- [47] Meejoung Kim. “Supervised learning-based DDoS attacks detection: Tuning hyperparameters”. In: *ETRI Journal* 41.5 (2019), pp. 560–573.
- [48] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (Sept. 2016). URL: <http://arxiv.org/abs/1609.02907>.

- [49] Deirdre M Kirke. “Gender clustering in friendship networks: some sociological implications”. In: *Methodological Innovations Online* 4.1 (2009), pp. 23–36.
- [50] Mikko Kivelä et al. “Multilayer networks”. In: *Journal of complex networks* 2.3 (2014), pp. 203–271.
- [51] Thijs Kooi et al. “Large scale deep learning for computer aided detection of mammographic lesions”. In: *Medical image analysis* 35 (2017), pp. 303–312.
- [52] Alexandra Krallman, Mark J Pelletier, and Frank G Adams. “@ Size vs.# Impact: Social media engagement differences amongst Facebook, Twitter, and Instagram”. In: *Celebrating America’s Pastimes: Baseball, Hot Dogs, Apple Pie and Marketing?* Springer, 2016, pp. 557–561.
- [53] Peter Laflin et al. “Discovering and validating influence in a dynamic online social network”. In: *Social Network Analysis and Mining* 3.4 (2013), pp. 1311–1323.
- [54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
- [55] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [56] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. <http://snap.stanford.edu/data>. June 2014.
- [57] Yuening Li et al. “PyODDS: An end-to-end outlier detection system with automated machine learning”. In: *Companion Proceedings of the Web Conference 2020*. 2020, pp. 153–157.
- [58] Yuening Li et al. “Specac: Spectral autoencoder for anomaly detection in attributed networks”. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 2233–2236.

- [59] Bin Liu, Zhisheng Yan, and Chang Wen Chen. “Medium access control for wireless body area networks with QoS provisioning and energy efficient design”. In: *IEEE transactions on mobile computing* 16.2 (2016), pp. 422–434.
- [60] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. “AUC: a misleading measure of the performance of predictive distribution models”. In: *Global ecology and Biogeography* 17.2 (2008), pp. 145–151.
- [61] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).
- [62] Kaushalya Madhawa and Tsuyoshi Murata. “Active Learning for Node Classification: An Evaluation”. In: *Entropy* 22.10 (2020), p. 1164.
- [63] Vijay Mahadevan et al. “Anomaly detection in crowded scenes”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1975–1981.
- [64] Alexander V Mantzaris et al. “Adaptive network diagram constructions for representing big data event streams on monitoring dashboards”. In: *Journal of Big Data* 6.1 (2019), pp. 1–19.
- [65] Victor Martinez, Carlos Cano, and Armando Blanco. “ProphNet: a generic prioritization method through propagation of information”. In: *BMC bioinformatics* 15.1 (2014), pp. 1–13.
- [66] Julian McAuley et al. “Image-based recommendations on styles and substitutes”. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 2015, pp. 43–52.
- [67] Andrew Kachites McCallum et al. “Automating the construction of internet portals with machine learning”. In: *Information Retrieval* 3.2 (2000), pp. 127–163.

- [68] Miller McPherson, Lynn Smith-Lovin, and James M Cook. “Birds of a feather: Homophily in social networks”. In: *Annual review of sociology* 27.1 (2001), pp. 415–444.
- [69] Emmanuel Müller et al. “Ranking outlier nodes in subspaces of attributed graphs”. In: *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*. IEEE. 2013, pp. 216–222.
- [70] Mark Newman. *Networks*. Oxford university press, 2018.
- [71] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.
- [72] Hoang NT and Takanori Maehara. “Revisiting graph neural networks: All we have is low-pass filters”. In: *arXiv preprint arXiv:1905.09550* (2019).
- [73] Ekaterina Olshannikova et al. “Conceptualizing big social data”. In: *Journal of Big Data* 4.1 (2017), pp. 1–19.
- [74] Felix L Opolka et al. “Spatio-temporal deep graph infomax”. In: *arXiv preprint arXiv:1904.06316* (2019).
- [75] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web*. Tech. rep. Stanford InfoLab, 1999.
- [76] Akash Sampurnanand Pandey and KK Shukla. “Application of bayesian automated hyperparameter tuning on classifiers predicting customer retention in banking industry”. In: *Data Management, Analytics and Innovation*. Springer, 2021, pp. 83–100.
- [77] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

- [78] Bryan Perozzi et al. “Focused clustering and outlier detection in large attributed graphs”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 1346–1355.
- [79] Phuong Pho and Alexander V Mantzaris. “Regularized Simple Graph Convolution (SGC) for improved interpretability of large datasets”. In: *Journal of Big Data* 7.1 (2020), pp. 1–17.
- [80] Marco AF Pimentel et al. “A review of novelty detection”. In: *Signal Processing* 99 (2014), pp. 215–249.
- [81] Tahereh Pourhabibi et al. “Fraud detection: A systematic literature review of graph-based anomaly detection approaches”. In: *Decision Support Systems* 133 (2020), p. 113303.
- [82] Filippo Radicchi, Santo Fortunato, and Alessandro Vespignani. “Citation networks”. In: *Models of science dynamics*. Springer, 2012, pp. 233–257.
- [83] G Thippa Reddy et al. “Analysis of dimensionality reduction techniques on big data”. In: *IEEE Access* 8 (2020), pp. 54776–54788.
- [84] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.
- [85] Yu Rong et al. “Dropege: Towards deep graph convolutional networks on node classification”. In: *arXiv preprint arXiv:1907.10903* (2019).
- [86] Andrea Rossi et al. “Knowledge graph embedding for link prediction: A comparative analysis”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.2 (2021), pp. 1–49.
- [87] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. “Multi-scale attributed node embedding”. In: *Journal of Complex Networks* 9.2 (2021), cnab014.

- [88] Benedek Rozemberczki and Rik Sarkar. “Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models”. In: *Proceedings of the 29th ACM international conference on information & knowledge management*. 2020, pp. 1325–1334.
- [89] Lukas Ruff et al. “Deep one-class classification”. In: *International conference on machine learning*. PMLR. 2018, pp. 4393–4402.
- [90] Aliaksei Sandryhaila and José MF Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.
- [91] Franco Scarselli et al. “The graph neural network model”. In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.
- [92] J Ben Schafer et al. “Collaborative filtering recommender systems”. In: *The adaptive web*. Springer, 2007, pp. 291–324.
- [93] Catharina Schmidt et al. “Generation “always on” turned off. Effects of smartphone separation on anxiety mediated by the fear of missing out”. In: *International Conference on Human-Computer Interaction*. Springer. 2018, pp. 436–443.
- [94] Prithviraj Sen et al. “Collective classification in network data”. In: *AI magazine* 29.3 (2008), pp. 93–93.
- [95] Burr Settles. “Active learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 6.1 (2012), pp. 1–114.
- [96] Burr Settles. “Active learning literature survey”. In: (2009).
- [97] Burr Settles and Mark Craven. “An analysis of active learning strategies for sequence labeling tasks”. In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 2008, pp. 1070–1079.
- [98] Oleksandr Shchur et al. “Pitfalls of Graph Neural Network Evaluation”. In: *Relational Representation Learning Workshop, NeurIPS 2018* (2018).

- [99] Nasrullah Sheikh, Zekarias Kefato, and Alberto Montresor. “gat2vec: representation learning for attributed graphs”. In: *Computing* 101.3 (2019), pp. 187–209.
- [100] Kai Shu et al. “User identity linkage across online social networks: A review”. In: *Acm Sigkdd Explorations Newsletter* 18.2 (2017), pp. 5–17.
- [101] David I Shuman et al. “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.
- [102] Aditya Siddhant and Zachary C. Lipton. “Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2904–2909. DOI: 10.18653/v1/D18-1318. URL: <https://aclanthology.org/D18-1318>.
- [103] Felipe Bonow Soares, Raquel Recuero, and Gabriela Zago. “Influencers in polarized political networks on Twitter”. In: *Proceedings of the 9th international conference on social media and society*. 2018, pp. 168–177.
- [104] Jieun Son and Seoung Bum Kim. “Academic paper recommender system using multilevel simultaneous citation networks”. In: *Decision Support Systems* 105 (2018), pp. 24–33.
- [105] Xiaoyuan Su and Taghi M Khoshgoftaar. “A survey of collaborative filtering techniques”. In: *Advances in artificial intelligence* 2009 (2009).
- [106] Shan Suthaharan. “Big data classification: Problems and challenges in network intrusion prediction with machine learning”. In: *ACM SIGMETRICS Performance Evaluation Review* 41.4 (2014), pp. 70–73.

- [107] Min Tang, Xiaoqiang Luo, and Salim Roukos. “Active learning for statistical natural language parsing”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 2002, pp. 120–127.
- [108] Xin-Min Tao et al. “A novel model of one-class bearing fault detection using SVDD and genetic algorithm”. In: *2007 2nd IEEE Conference on Industrial Electronics and Applications*. IEEE. 2007, pp. 802–807.
- [109] Troy Tassier. “Labor market implications of weak ties”. In: *Southern Economic Journal* (2006), pp. 704–719.
- [110] David MJ Tax and Robert PW Duin. “Support vector data description”. In: *Machine learning* 54.1 (2004), pp. 45–66.
- [111] Nguyen Thai-Nghe et al. “Recommender system for predicting student performance”. In: *Procedia Computer Science* 1.2 (2010), pp. 2811–2819.
- [112] Marina Thottan and Chuanyi Ji. “Anomaly detection in IP networks”. In: *IEEE Transactions on signal processing* 51.8 (2003), pp. 2191–2204.
- [113] Yonglong Tian et al. “Deep learning strong parts for pedestrian detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1904–1912.
- [114] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.
- [115] Lyle H Ungar and Dean P Foster. “Clustering methods for collaborative filtering”. In: *AAAI workshop on recommendation systems*. Vol. 1. Menlo Park, CA. 1998, pp. 114–129.
- [116] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

- [117] Jianlin Wang et al. “Dynamic hypersphere based support vector data description for batch process monitoring”. In: *Chemometrics and Intelligent Laboratory Systems* 172 (2018), pp. 17–32.
- [118] Minjie Wang et al. “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks”. In: *arXiv preprint arXiv:1909.01315* (2019).
- [119] Peng Wang et al. “Link prediction in social networks: the state-of-the-art”. In: *Science China Information Sciences* 58.1 (2015), pp. 1–38.
- [120] Xuhong Wang et al. “One-class graph neural networks for anomaly detection in attributed networks”. In: *Neural Computing and Applications* (2021), pp. 1–13.
- [121] Felix Wu et al. “Simplifying graph convolutional networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6861–6871.
- [122] Yuexin Wu et al. “Active learning for graph neural networks via node feature propagation”. In: *arXiv preprint arXiv:1910.07567* (2019).
- [123] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* (2020).
- [124] Feng Xia et al. “Big scholarly data: A survey”. In: *IEEE Transactions on Big Data* 3.1 (2017), pp. 18–35.
- [125] Tian Xie and Jeffrey C Grossman. “Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties”. In: *Physical review letters* 120.14 (2018), p. 145301.
- [126] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810 . 00826 [cs.LG].

- [127] Zekun Xu, Deovrat Kakde, and Arin Chaudhuri. “Automatic Hyperparameter Tuning Method for Local Outlier Factor, with Applications to Anomaly Detection”. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 4201–4207.
- [128] Jian-Xiong Zhang et al. “Identifying a set of influential spreaders in complex networks”. In: *Scientific reports* 6 (2016), p. 27823.
- [129] Muhan Zhang and Yixin Chen. “Link prediction based on graph neural networks”. In: *Advances in neural information processing systems* 31 (2018).
- [130] Richong Zhang and Thomas Tran. “An information gain-based approach for recommending useful product reviews”. In: *Knowledge and Information Systems* 26.3 (2011), pp. 419–434.
- [131] Si Zhang et al. “Graph convolutional networks: a comprehensive review”. In: *Computational Social Networks* 6.1 (2019), p. 11.
- [132] Xiao-Meng Zhang et al. “Graph neural networks and their current applications in bioinformatics”. In: *Frontiers in Genetics* 12 (2021).
- [133] Qinghe Zheng et al. “Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process”. In: *IEEE Access* 6 (2018), pp. 15844–15869.
- [134] Qinghe Zheng et al. “Spectrum interference-based two-level data augmentation method in deep learning for automatic modulation classification”. In: *Neural Computing and Applications* (2020), pp. 1–23.
- [135] Jie Zhou et al. “Graph neural networks: A review of methods and applications”. In: *arXiv preprint arXiv:1812.08434* (2018).
- [136] Jie Zhou et al. “Graph neural networks: A review of methods and applications”. In: *AI Open* 1 (2020), pp. 57–81.