Graph Convolutional Neural Networks with Edge-Node Switching

by

Xiaodong Jiang

(Under the Direction of Sheng Li)

Abstract

In this dissertation, we present *CensNet*, **C**onvolution with **E**dge-**N**ode **S**witching graph neural network, for semi-supervised classification and regression in graph-structured data with both node and edge features. CensNet is a general graph embedding framework, which embeds both nodes and edges to a latent feature space. By using *line graph* of the original undirected graph, the role of nodes and edges are switched, and two novel graph convolution operations are proposed for feature propagation. Experimental results on real-world academic citation networks and quantum chemistry graphs show that our approach has achieved or matched the state-of-the-art performance.

Index words: Deep learning, Statistical graph analysis, Relational learning, Semi-supervised learning

Graph Convolutional Neural Networks with Edge-Node Switching

by

Xiaodong Jiang

B.S., Beijing University of Technology, 2014

M.S., University of Georgia, 2016

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2019

Graph Convolutional Neural Networks with Edge-Node Switching

by

Xiaodong Jiang

Approved:

Major Professor: Sheng Li

Committee: Pengsheng Ji
Jaewoo Lee

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2019

Graph Convolutional Neural Networks with Edge-Node Switching

Xiaodong Jiang

May 2019

For Mom and Dad

Acknowledgments

I would like to thank my major advisor, Dr. Sheng Li, for his continuous, selfless, and endless help, support, and guidance, this dissertation cannot be finished without him. Dr. Li introduced me to the field of deep learning, and he always encourages me when I get lost, confused, and uninspired in my research. He is my role model researcher in the filed of computer science - knowledgeable, determined, passionate, and hard working. I would also thank my committee members, Dr. Pengsheng Ji and Dr. Jaewoo Lee, for the participation and valuable discussions for this topic.

There is never an easy path that leads to the destination. I would like to thanks my parents for their love, encouragement, support, and providing me the best education in my life, without whom I will never enjoy so many opportunities.

Finally, I want to thank my wife, Xuan Zhang, who gives me the best time in my life. Without her accompany, I will never achieve so many goals in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning models like convolutional neural networks (CNN) have been remarkably successful in many domains LeCun et al. [2015], including computer vision, natural language processing, signal processing, etc. In particular, CNN and its variants are capable of extracting multi-scale localized spatial features and producing highly expressive representations Krizhevsky et al. [2012]; He et al. [2016]. Instead of using sophisticated feature engineering procedures that heavily rely on domain expertise, CNN based models are usually trained in an end-to-end fashion, and they can model the high-order interactions among input features for specific tasks. The convolution operations in CNN are well defined on data with underlying Euclidean structures (e.g., images and speech), but they cannot be directly generalized to non-Euclidean data such as graphs and manifolds Bronstein et al. [2017].

Graph-structured data is ubiquitous, from social network platforms to cita-

tion and co-authorship relations, from protein-protein interactions to chemical molecules. Graph, as a complex data structure, is very effective in describing the relationships (edges) of objects (nodes). Due to the expressive power and flexibility of graph-structured data, graph neural networks (GNN) have attracted increasing attention in recent years, which try to adapt the effective deep representation learning approaches from Euclidean to non-Eculidean domains Zhou et al. [2018]. The earliest GNN method might be traced back to the work in Scarselli et al. [2009], which extends the general neural networks to graph domain. Along this research direction, many other GNN models have been proposed recently, such as the ChebNet Defferrard et al. [2016b], graph convolutional networks (GCN) Kipf and Welling [2017], GraphSAGE Hamilton et al. [2017], Lanczosnet Liao et al. [2019], etc. By leveraging the node adjacency matrix, these GNN models analogously define convolution operators on graphs in either spectral or spatial spaces and have obtained promising performance in tasks like node classification Zhou et al. [2018]. With a few notable exceptions Monti et al. [2017]; Velikovi et al. [2018]; Schlichtkrull et al. [2018], GNN methods mainly focus on obtaining effective node embeddings, but ignore the information associated with edges that can be beneficial to many tasks such as node or edge classification, link prediction, community detection, and regression.

In this paper, we aim to learn both node embeddings and edge embeddings for graphs. Graphs in reality usually contain both node and edge features. In social networks, the node features could be demographic information or user behaviors, while the edge features might be the type of relationships or the years of friendship.

In citation networks, the node features could be document-level embeddings of papers, and the edge features might indicate the common keywords or co-authors of two articles. In chemistry domain, each compound can be considered as a graph, where atoms are nodes, and properties of chemical bonds are edge features. We justify the motivation of jointly learning node and edge embeddings from the following two aspects. First, it is clear that edge and nodes always provide complementary feature information, which will be helpful for graph embedding. Second, learning edge embeddings is essential for edge-relevant tasks, such as edge classification and regression.

Inspired by the *Line Graph* in graph theory Harary and Norman [1960], we propose a novel convolution with edge-node switching network (CensNet) for learning node and edge embeddings. Let G denote the node adjacency matrix of a graph, its line graph $L(G)$ can be constructed to represent the adjacencies between edges of G. In our framework, the role of node and edge can be switched, and CensNet conducts the graph convolution operations on both the input graph G and its line graph counterpart. With the help of node and edge features, CensNet employs two forward-pass feature propagation rules on G and $L(G)$ to alternatively update the node and edge embeddings. Also, a mini-batch training algorithm for CensNet is devised to handle large-scale graphs.

The main contributions of this work are summarized below.

- **Co-embedding of nodes and edges.** The proposed CensNet is a general graph embedding framework which can embed both nodes and edges to a latent feature space simultaneously, with the help of graph structure

and node/edge features. Moreover, the node and edge embeddings can be mutually enriched owing to the alternative update rules.

- **Convolution on line graph.** We design convolution operations on both the input graph and its line graph counterpart, where the role of node and edge are switched.
- **Diverse learning tasks on graphs.** We apply CensNet to several graph-based learning tasks, including multi-task graph classification, graph regression, and semi-supervised node classification.
- **Extensive evaluations on benchmark data sets.** The extensive experiments on citation networks (Cora, Citeseer Sen et al. [2008], and PubMed Namata et al. [2012]) and Quantum Chemistry data sets (*Tox21* and *Lipophilicity* Wu et al. [2018]) show that our method has achieved or matched state-of-the-art performance.

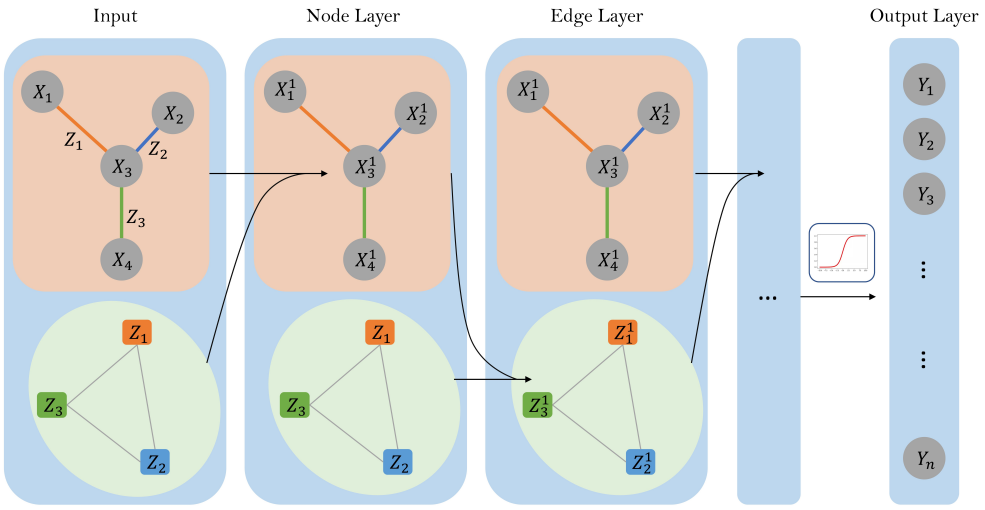


Figure 1.1: Illustrative depiction of Convolution with Edge-Node Switching (CensNet). The upper (orange color) components are convolution operations on node adjacency matrix and node features, while the lower (green color) components are the corresponding line graph convolution. Two types of layers as a combo, (1). Node Layer, update the node embedding with node and edge embedding from the previous layer, and (2). Edge Layer, update the edge embedding with the edge and node embedded features from the preceding layer.

Chapter 2

Preliminaries

2.1 Notations

We present the mathematical notations for a graph with node and edge features in the following numbered list.

- We assume an undirected graph $G = \{V, E\}$ with node set V and edge set E, and $N_v = |V|$ and $N_e = |E|$ denote the number of nodes and edges, respectively.

- Let $A_v \in \mathbb{R}^{N_v \times N_v}$ be the adjacency matrix of G, where each element $A_v(i, j)$ denotes the connectivity of node i and node j, where $i, j \in \{1, 2, ..., N_v\}$. A_v is a binary matrix in unweighted graph.

- Let $X \in \mathbb{R}^{N_v \times d_v}$ be the node feature matrix, where each node is associated with a d_v-dimensional feature vector.

- Let $A_e \in \mathbb{R}^{N_e \times N_e}$ be the binary edge adjacency matrix of G, or node adjacency matrix of $L(G)$. $A_e(m, n)$ equals to 1 if the edge m and edge n are connected by a node in G, otherwise 0, where $m, n \in \{1, 2, ..., N_e\}$.

- Let $Z \in \mathbb{R}^{N_e \times d_e}$ denote the edge feature matrix, where each edge has a d_e-dimensional feature vector.

- Let $H_v^{(l)}$ be the l-th hidden layer of node convolution with $H_v^{(0)} = X$; Define $H_e^{(l)}$ as the l-the layer of edge convolution, and $H_e^{(0)} = Z$.

We will also introduce other notations in the rest of the paper when necessary.

2.2 Graph Convolution and Embedding

Given a graph G and its corresponding node feature matrix X, there are two major graph convolutions in literature, *spectral* convolution in the Fourier domain and *spatial* convolution in the node (or vertex) domain. We discard the subscripts of our notations for a moment, assuming X is the node feature matrix, and A is the node adjacency matrix.

Spectral graph convolution. A spectral graph convolution is defined as the multiplication of a signal with a filter in the Fourier domain of the graph. The graph Fourier transform Y is defined as the multiplication of a graph signal (e.g., node features X) with the eigenvector matrix U of the graph Laplacian L, i.e., $Y = U^T X$ and $U \Sigma U^T = L$. The graph Laplacian L can be defined in different ways: the simple Laplacian $D - A$, the symmetric normalized Laplacian $I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$,

or the random walk Laplacian $I - D^{-1}A$, where I is an identity matrix and D is the diagonal degree matrix. The symmetric normalized Laplacian is usually desirable due to the nice properties including symmetric, positive-semidefine, and all eigenvalues are in $[0, 2]$. The ChebyNet Defferrard et al. [2016b] and GCN Kipf and Welling [2017] are based on spectral graph convolutions. The GCN proposes a layer-wise propagation rule based on an approximated graph spectral kernel as follows

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

where $\tilde{A} = A + I$ is the adjacency matrix with self-connections, $\tilde{D}$ is the degree matrix, $H^{(l)}$ and $W^{(l)}$ are the hidden feature matrix and learnable weight in the l-th layer.

Spatial graph convolution. A spatial graph convolution is defined on the node domain, which can integrate or aggregate the signals among its neighbor nodes. MoNet Wu et al. [2018] and GraphSAGE Hamilton et al. [2017] are aggregation based representation learning models in this direction.

Node and Edge Embeddings. There have only been a few scattered examples that can link both node and edge features in a graph convolution simultaneously, which can be categorized as two approaches. The first one is to define different weight matrices for each relation or dimension on the edge features and aggregate different relation propagation in an additive fashion. Schlichtkrull et al.

[2018] used the following rule for the forward-pass update

$$h_i^{l+1} = \sigma(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)})$$

where each relation $r \in \mathcal{R}$ has its corresponding weight matrix W_r. Such a simple aggregation enjoys the efficiency in computation but cannot capture the dependence of different relations and the interaction between the node and edge features. The second approach is to add the attention mechanism to graph convolutions by specifying different weights to different nodes in a neighborhood. MonNet Monti et al. [2017] and GAT Velikovi et al. [2018] are examples along this line. More importantly, both of these two approaches can only handle discrete or low-dimensional edge features.

Different from existing graph convolution and embedding methods, the proposed CensNet framework is independent of the choice of graph convolution, although we only implement one representative spectral-based method Kipf and Welling [2017] in this paper. Moreover, the co-embedding of nodes and edges in CensNet is inspired by *line graph* and driven by mutually-enriched feature propagation rules, which can handle high-dimensional discrete or continuous node/edge features.

Chapter 3

Methodology

The CensNet framework consists of two types of layers, node layer and edge layer. Figure 1.1 shows the CensNet architecture for semi-supervised node classification. The input layer comprises of a node adjacency matrix and the corresponding node features, as well as its line graph counterpart - the edge adjacency matrix and edge features. The three colored edges (i.e., z_1, z_2 and z_3) in the sample graph are converted to three line graph nodes (squared shape). We define an CensNet combo as two types of layers, node layer and edge layer. In the node layer, all input data are processed to update the node embedding, while keeping the line graph (edge adjacency matrix and edge features) flow forward without any change. In the edge layer, we combine the updated node embedding with the line graph to update the edge embedding. Depending on the specific task and the availability of labels, CensNet adopts different types of activation functions for the node or edge embedding matrices. For example, in the graph node semi-classification task

(e.g., paper classification in citation networks), we have the label for each node, and thus we can use the sigmoid function for the node embedding matrix in the final layer. For the graph classification or regression task (e.g., the molecular property prediction), we may apply an average pooling layer to reshape the node embeddings and obtain graph-level embeddings.

We should note, the CensNet framework is a high-level abstract of the interactive graph embedding with both nodes and edges. One does not necessarily use the approximated spectral graph kernel for the convolution. Other new techniques such as the kernels in Lanczos Network Liao et al. [2019] can also be applied to our framework.

3.1 Propagation Rules

The CensNet uses approximated spectral graph convolution in the layer-wise propagation. We define the normalized (Laplacianized) node adjacency matrix with self-loop as

$$\tilde{A}_v = D_v^{-\frac{1}{2}}(A_v + I_{N_v})D_v^{-\frac{1}{2}} \tag{3.1}$$

where D_v is the diagonal degree matrix of $A_v + I_{N_v}$, and I is an identity matrix.

Propagation rule for node layer. The layer-wise propagation rule for node feature in the $(l+1)$-th layer is defined as

$$H_v^{(l+1)} = \sigma(T\Phi(H_e^{(l)}P_e)T^T \odot \tilde{A}_v H_v^{(l)} W_v) \tag{3.2}$$

where the matrix $T \in \mathbb{R}^{N_v \times N_e}$ is a binary transformation matrix and $T_{i,m}$ represents whether edge m connects node i. Given the fact that each edge is formed by two nodes, each column of matrix T has two elements being 1 and all others are 0. P_e is a d_e-dimensional vector, defined as the learnable weights for edge feature vectors. Φ denotes the diagonalization operation, which places a one dimensional vector to the diagonals of a square matrix. $\odot$ denotes the Hadamard product or element-wise product. Another view of this rule is to map corresponding element from $T\Phi(H_e^{(l)}P_e)T^T$ to the normalized node adjacency matrix, and the $T\Phi(H_e^{(l)}P_e)T^T \odot \tilde{A}_v$ is a fused node adjacency matrix by using information from the line graph counterpart. The line graph brings zero impact if there is no physical edge, and thus we maintain the sparsity of the original graph, bringing significant computing benefits.

Another notable and interesting character of CensNet is that it degrades to the regular GCN when the edge feature is a scalar 1. In this case, the first component in Equation (3.2) (i.e., $T\Phi(H_e^{(l)}P_e)T^T$) reduces to an identity matrix and the remaining part $\tilde{A}_v H_v^{(l)} W_v$ is exactly the propagation rule defined in Kipf and Welling [2017].

Propagation rule for edge layer. Similarly, the normalized (Laplacianized) edge adjacency matrix is defined as

$$\tilde{A}_e = D_e^{-\frac{1}{2}}(A_e + I_{N_e})D_e^{-\frac{1}{2}}, \tag{3.3}$$

where D_e is the degree matrix of $A_v + I_{N_v}$.

Furthermore, we define the propagation rule for edge features as follows

$$H_e^{(l+1)} = \sigma(T^T \Phi(H_v^{(l)} P_v) T \odot A_e H_e^{(l)} W_e). \tag{3.4}$$

The T matrix is same as Equation (3.2), while P_v represents the learnable weight for the nodes. As a return, the node feature and adjacency matrix are used to improve the edge embedding. These two components bridge signals on nodes and edges, and the node and edge embeddings are updated alternatively.

3.2 Task-Dependent Loss Functions

The designs of the output layer, as well as the loss function, are task dependent. For node or edge classification tasks, we may apply the sigmoid function to the final hidden node or edge layers; for the graph classification task, we need an extra pooling operation that maps the node-level embeddings to a graph-level representation. For example, we take the average prediction of the atoms in a molecule as the graph-level output. The graph-level max pooling might not be appropriate from a practical point of view; a learnable parameter (weight) for each node is also not acceptable because of the tendency to overfitting.

For the semi-supervised node classification task, the loss function is defined as

$$\mathcal{L}(\Theta) = -\sum_{l \in Y_L} \sum_{f=1}^{F} Y_{lf} \log M_{lf} \tag{3.5}$$

where Y_L is a set of nodes with labels, and M is the softmax results of the output

from the last node layer, assuming the final node feature map has F dimensions. The $\Theta = (W_v, W_e, P_v, P_e)$ is the parameter set. For multi-task graph classification, we may sum up the cross entropy from each target.

For regression task where the response is a continuous variable, we define the following loss function by using the l^p regularized mean square error (MSE)

$$\mathcal{L}(\Theta) = \sum_{l \in Y_L} \sum_{f=1}^{F} ||Y_{lf} - \hat{Y}_{lf}||_2^2 + \lambda ||\Theta||_p \tag{3.6}$$

where $\hat{Y}$ is the predicted outcome from the last node hidden layer. The regularized term is used to control the model complexity and avoid overfitting.

3.3 Training Algorithms

We show our algorithm for semi-supervised node classification in *Algorithm 1*, which uses the layer-wise propagation rules defined in Equations (3.2) and (3.4). Optimization algorithms, such as Adam or SGD, are employed to deal with the cross-entropy loss function. When the input data cannot fit into the GPU memory, a mini-batch strategy is usually preferred Kingma and Ba [2014]. Thus, we also design a mini-batch training algorithm for large graphs in *Algorithm 2* for the node classification task with few labeled nodes. The key idea is to sample the nodes from training, validation, and test set proportionally, to construct the batches. Our empirical results show that such a training strategy could match the performance of training with the entire graph.

Algorithm 1 CensNet for Node Classification

Input: *Node adjacency matrix* A_v
node feature matrix X
node label Y
edge feature matrix Z
nonlinearity σ

Build *transformation matrix* T and *edge adjacency matrix* A_e

Run Adam to minimize $\mathcal{L}$

$$W_v, W_e, P_v, P_e := \arg\min_{W_v, W_e, P_v, P_e} \mathcal{L}$$

Output: $\mathrm{softmax}(H_v)$

Algorithm 2 Mini-batch CensNet

for each epoch **do**
    Construct batches to make each batch containing subgraph $G' \in G$ and the nodes in G' are proportionally selected from training, validation, and testing sets.
    for each batch **do**
        Run Algorithm 1
    end for
end for

Chapter 4

Experiments

We evaluate the proposed CensNet method for three tasks on five benchmark data sets.

4.0.1 Data Description and Preprocessing

Tox21. The *Toxicology in the 21st Century (Tox21, Wu et al. [2018])* initiative created a public database measuring toxicity of compounds, which has been used in the 2014 *Tox21* Data Challenge. This dataset contains quantitative toxicity measurements for 7,831 environmental compounds and drugs. A compound structure (in SMILE format) is usually expressed as a graph with atoms as nodes and bonds being edges. There are 55 bond features, and each atom has 25 features. Each compound is associated with 12 binary labels that represent the outcome (active/inactive) of 12 different toxicological experiments. There are about 20% missing values in these labels, and we exclude those observations when computing

the loss but still keep them in the training process.

Lipophilicity. The *Lipophilicity* is an important feature of drug molecules that affects both membrane permeability and solubility Wu et al. [2018]. This dataset provides experimental results of octanol/water distribution coefficient (logD at pH 7.4) of 4,200 compounds. There are 34 features on the edge (bond) and 25 features on the node (atom), while the response is one single continuous variable.

We follow the preprocessing steps in Wu et al. [2018] to convert each compound in *Tox21* and *Lipophilicity* to a small graph, and remove the compounds whose SMILE representation cannot be converted to a graph structure. We then randomly split the dataset to different partitions as training, validation and test sets, respectively. We consider 4 data splitting settings, by keeping 60%, 70%, 80%, and 90% of the molecule graphs as training set, while equally breaking the rest of the data sets as validation and test sets.

Cora, Citeseer, and PubMed. These three data sets are benchmarks for the semi-supervised node classification task, which has been analyzed by many graph convolutional network models such as the ones in Defferrard et al. [2016b]; Kipf and Welling [2017]; Hamilton et al. [2017]; Velikovi et al. [2018]; Liao et al. [2019]. *Cora* has 2,708 nodes (papers) and 5,429 edges (citation links), and each node has 1,433 tf-idf features. The papers are classified to 7 different research areas thus the response has seven different values. *Citeseer* has 3,327 nodes and 4,732 edges with 3,703 node features; the papers are grouped into 6 research fields. *PubMed* contains 19,717 nodes and 44,338 edges, each node has 500 features and the papers are in 3 categories. Please remark that these citation graphs are not

naturally good benchmarks for CensNet because there is no available edge feature. However, we still run our algorithm with effortless hand-crafted edge features to show the competitive performance of CensNet. Examples of such edge features could be the pairwise node feature correlations or cosine similarities.

For three citation graphs, we create two simple edge features: (1) the pairwise cosine similarities between corresponding node features, and (2) a 2-dimensional vector to represent the edge directions. If paper A cites paper B then the vector is $[1, 0]$, otherwise $[0, 1]$. We follow the splitting strategy in Liao et al. [2019] and implement the experiments for different label rate. We evaluate our method and baselines with 3%, 1% and 0.5% labeled data in training set on *Cora*, 1%, 0.5% and 0.3% on *Citeseer*, and 0.1%, 0.05% and 0.03% on *PubMed*. For all three data sets, we random select 50% for validation and the rest for testing.

4.1 Experimental Settings

Three tasks are implemented in our experiments.

Multi-task graph classification. We evaluate the performance of CensNet on the *Tox21* dataset under 4 data splitting scenarios. Wu et al. [2018] presented a comprehensive model comparisons for the *Tox21* data, and the state-of-the-art method is GCN Kipf and Welling [2017]. We implemented the minibatch GCN, and other two classical classification algorithms, logistic regression and random forest, as three baselines. We report the area under the ROC curve (AUC) metric in both validation and test data sets for all compared methods.

Graph regression. The regression task is similar to graph classification but using different loss function and performance metric. We use the same data splitting strategies as in *Tox21*. The baseline algorithms are GCN, linear regression, and random forest regression. We also report the root mean square error (RMSE) in both validation and test sets.

Semi-supervised node classification. This is a classical task in graph learning and statistical learning communities, and the most widely used benchmark data sets are three citation networks, *Cora*, *Citeseer* and *PubMed*. We adopt a few-shot learning strategy, i.e., only keep a small number of labeled data in training set while splitting the rest of the data for validation and test. We compare CensNet with seven representative graph convolution networks when using different percentages of labeled data. The seven baselines include ChebyNet Defferrard et al. [2016a], GCN Kipf and Welling [2017], GraphSAGE Hamilton et al. [2017], GAT Velikovi et al. [2018], LNet and AdaLet Liao et al. [2019]. To the best of our knowledge, LNet and AdaLet are the state-of-the-art methods for this task Liao et al. [2019], and we will present a comprehensive comparison between these two methods and the proposed CensNet.

All experiments are conducted on an Azure Linux VM (CPU: Intel(R) Xeon(R) CPU E5-2690 v3, GPU: NVIDIA Tesla K80). We implemented all graph convolution network algorithms in PyTorch Paszke et al. [2017] v1.0.0. For other classical algorithms (random forest, linear regression, logistic regression), we used the implementations in the Python package Scikit-learn Pedregosa et al. [2011]. For graph convolution models, we didn't implement any sophisticated fine-tuning strategies

but tried different settings of learning rate from $\{0.01, 0.005, 0.001, 0.0005\}$, batch size $\{16, 32, 64, 128, 256\}$, number of epochs $\{200, 300, 500, 1000\}$, etc. We report the best-performed results for each algorithm. For tree-based methods, we use cross-validation to tune the parameters; for linear and logistic regression models, we run the algorithms without using variable selection.

4.2 Results and Discussions

Multi-task graph classification. Table 4.1 shows the AUC values on the validation and test sets, with four different training label rates. We replicate all experiments three times and report the mean and standard deviation of AUC values. The CensNet algorithm maintains significant advantages over all other methods in all settings, while both GCN and CensNet perform better than the other two traditional methods. The logistic regression and random forest models can hardly capture the association between signals and response, even with increased training sets. Figure 4.1(a) shows the changes in validation AUC in 200 epochs. The GCN's curve becomes flat within 20 epochs while the CensNet can continuously improve. Also, the CensNet with fewer training set can beat the GCN with more training data, uniformly and consistently.

Graph regression. Compared with three baseline methods, our CensNet has achieved the best performance in both validation and test sets under four training settings. For a fair comparison, we replicate all experiments three times and report the mean RMSE with standard errors. The following conclusions can be

summarized from Table 4.1: (1). all these 4 algorithms achieve better performance with larger training sets, while random forest and logistic regression only obtain limited performance lift. (2). both GCN and CensNet have large positive margins over traditional methods, indicating that the graph structure is not neglectable in the molecule regression task. (3). CensNet improves the performance of GCN by 5% - 15% in RMSE, which implies that considering the edge features in the molecule can improve the quality of node embedding, as a consequence, bring significant benefits to the learning process. Figure 4.1(b) shows that the CensNet gains leading positions after around 50-75 epochs, while GCN's curves still keep flat.

Train PCT	Data Split	*Tox21* (AUC)				*Lipophilicity* (RMSE)			
		RF	Logistic	GCN	CensNet	RF	LR	GCN	CensNet
60%	Val.	0.69±0.01	0.69±0.01	0.72±0.00	**0.76±0.00**	1.19±0.01	1.46±0.37	1.04±0.01	**0.94±0.01**
	Test	0.71±0.01	0.71±0.01	0.73±0.00	**0.77±0.00**	1.16± 0.02	1.17 ±0.03	1.06±0.00	**0.97±0.01**
70%	Val.	0.70 ± 0.01	0.70±0.01	0.73±0.00	**0.76±0.00**	1.18±0.02	1.19±0.01	1.02±0.01	**0.92 ±0.01**
	Test	0.70 ± 0.01	0.71±0.01	0.74±0.00	**0.77±0.00**	1.16±0.02	1.17±0.04	1.05 ±0.01	**0.95±0.01**
80%	Val.	0.71±0.01	0.71±0.01	0.72±0.00	**0.76±0.00**	1.17±0.02	1.16±0.02	1.05±0.01	**0.96±0.01**
	Test	0.71±0.01	0.71±0.01	0.75±0.00	**0.78±0.00**	1.16±0.01	1.15±0.01	1.05±0.01	**0.93±0.01**
90%	Val.	0.71±0.02	0.71±0.01	0.74±0.00	**0.78±0.01**	1.18±0.02	1.18±0.03	1.08±0.00	**0.94±0.02**
	Test	0.71±0.02	0.71±0.02	0.75±0.00	**0.79±0.01**	1.13±0.03	1.13±0.03	0.97±0.00	**0.83±0.02**

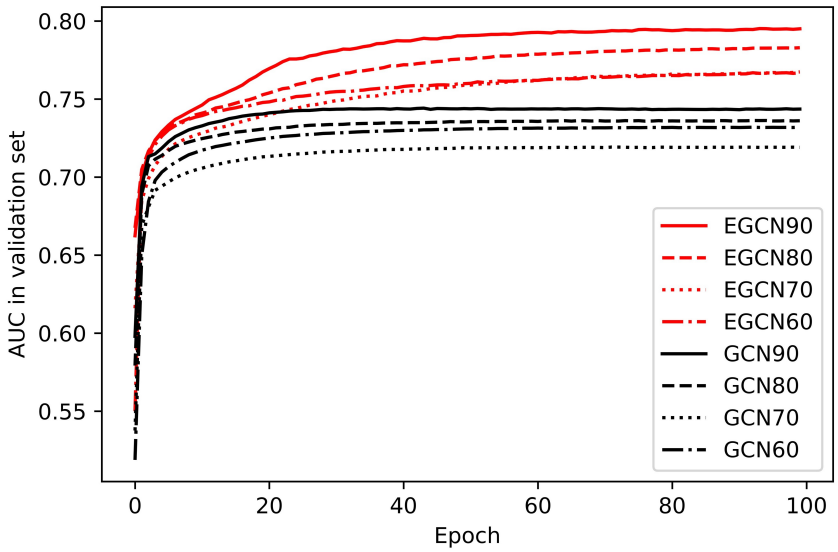
Table 4.1: Experimental results on *Tox21* and *Lipophilicity* data sets.

Semi-supervised node classification. We follow the same experimental settings in Liao et al. [2019] and re-use the benchmark results as our baselines. For all the experiments, we observed significant overfitting with low label rates for all graph convolution networks. We highlight the best-performed method (with the highest accuracy in the test set) in each setting for all three data sets. Our CensNet method obtains the best accuracy in 4 out of 9 experiments, which is followed by LNet and AdaLNet - the state-of-the-art algorithms for this task. One

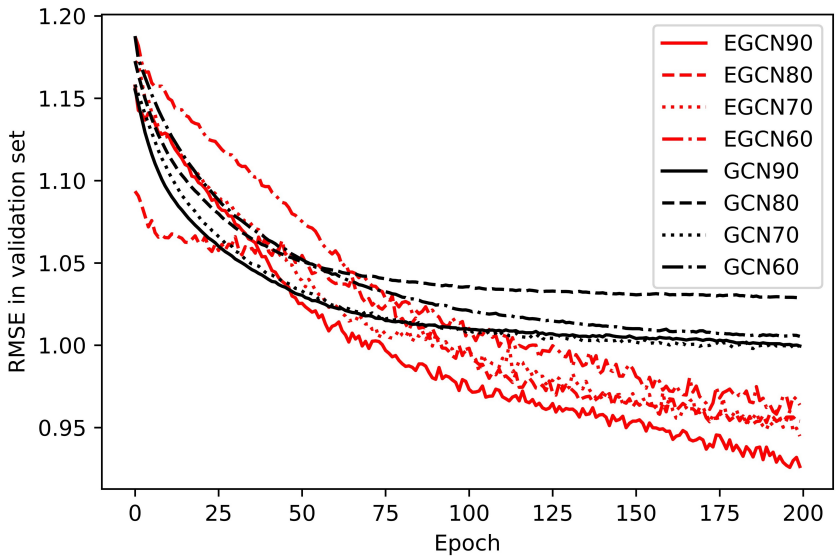
Data	TrainPCT	ChebyNet	GCN	GraphSAGE	GAT	LNet	AdaLNet	CensNet (Ours)
Cora	3%	62.1±6.7	74.0±2.8	64.2±4.0	56.8±7.9	76.3±2.3	77.7±2.4	**79.4±1.0**
	1%	44.2±5.6	61.0±7.2	49.0±5.8	48.6±8.0	66.1±8.2	**67.5±8.7**	67.1±1.3
	0.5%	33.9±5.0	52.9±7.4	37.5±5.4	41.4±6.9	58.1±8.2	**60.8±9.0**	57.7±3.9
Citeseer	1%	59.4±5.4	58.3±4.0	51.0±5.7	46.5±9.3	61.3±3.9	**63.3±1.8**	62.5±1.5
	0.5%	45.3±6.6	47.7±4.4	33.8±7.0	38.2±7.1	53.2±4.0	53.8±4.7	**57.6±3.0**
	0.3%	39.3±4.9	39.2±6.3	25.7±6.1	30.9±6.9	44.4±4.5	46.7±5.6	**49.4±3.6**
PubMed	0.1%	55.2±6.8	73.0±5.5	65.4±6.2	59.6±9.5	**73.4±5.1**	72.8±4.6	69.9±2.1
	0.05%	48.2±7.4	64.6±7.5	53.0±8.0	50.4±9.7	**68.8±5.6**	66.0±4.5	65.7±1.2
	0.03%	45.3±4.5	57.9±8.1	45.4±5.5	50.9±8.8	60.4±8.6	61.0±8.7	**61.4±2.8**

Table 4.2: Node classification accuracy (in percent) on three citation graph data sets.

may believe that adding edge information to the algorithm is not a fair comparison to the benchmarks; however, our newly created edge features are all from the benchmark data; thus no extra signal is introduced. The classical GCN Kipf and Welling [2017] also achieves competitive results in most scenarios, which coincides the conclusion from the extensive experiments in Shchur et al. [2018].



(a)



(b)

Figure 4.1: AUC/RMSE in validation set for *Tox21/Lipophilicity*. The name of each curve is formed with algorithm name and label ratio in training set. For example, CensNet90 means the CensNet algorithm with 90% data in training set.

Chapter 5

Conclusion

Graph convolution models open up an exciting area of deep learning in non-Euclidean space. Our newly proposed CensNet framework established a novel approach to learn the node and edge feature embeddings simultaneously. The extensive experiments on five benchmark data sets show that the proposed CensNet algorithm can achieve the state-of-art performance in three major graph learning tasks. We are currently exploring other graph kernels and more efficient training algorithms to scale up the deep graph learning to larger and more complex data sets.

Bibliography

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016a.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016b.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

Frank Harary and Robert Z. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 1960.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning

for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999134.2999257`.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 5 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BkedznAqKQ`.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and man-

ifolds using mixture model cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Galileo Mark Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Workshop on Mining and Learning with Graphs (MLG)*, 2012.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS-W*, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1): 61–80, Jan 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2005605.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI*

Magazine, 29(3):93–106, 2008. URL http://www.cs.iit.edu/~ml/pdfs/sen-aimag08.pdf.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

Petar Velikovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Zhenqin Wu, Bharath Ramsundar, EvanN. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018. doi: 10.1039/C7SC02664A. URL http://dx.doi.org/10.1039/C7SC02664A.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL http://arxiv.org/abs/1812.08434.