Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labeled Nodes

Ke Sun[1], Zhouchen Lin[2,5], Zhanxing Zhu[3,1,4*]

[1] Center for Data Science, Peking University, China
[2] Key Lab. of Machine Perception (MOE), School of EECS, Peking University, China
[3] School of Mathematical Science, Peking University, China
[4] Beijing Institute of Big Data Research (BIBDR)
[5] Samsung Research China - Beijing (SRC-B)
{ajksunke, zlin, zhanxing.zhu}@pku.edu.cn

Abstract

Graph Convolutional Networks (GCNs) play a crucial role in graph learning tasks, however, learning graph embedding with few supervised signals is still a difficult problem. In this paper, we propose a novel training algorithm for Graph Convolutional Network, called Multi-Stage Self-Supervised (M3S) Training Algorithm, combined with self-supervised learning approach, focusing on improving the generalization performance of GCNs on graphs with few labeled nodes. Firstly, a Multi-Stage Training Framework is provided as the basis of M3S training method. Then we leverage DeepCluster technique, a popular form of self-supervised learning, and design corresponding aligning mechanism on the embedding space to refine the Multi-Stage Training Framework, resulting in M3S Training Algorithm. Finally, extensive experimental results verify the superior performance of our algorithm on graphs with few labeled nodes under different label rates compared with other state-of-the-art approaches.

Introduction

With great expressive power, graphs have been employed as the representation of a wide range of systems across various areas, including social network (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017), physical systems (Battaglia et al. 2016; Sanchez-Gonzalez et al. 2018), protein-protein interaction networks (Hamaguchi et al. 2017) and knowledge graph (Fout et al. 2017). Recently, research of analyzing graphs with machine learning has been received more and more attention, mainly focusing on node classification (Kipf and Welling 2016), link prediction (Zhu, Song, and Chen 2016) and clustering tasks (Fortunato 2010).

Graph convolution can be regarded as the extension of standard convolution from Euclidean to non-Euclidean domain. Graph Convolutional Networks (GCNs) (Kipf and Welling 2016) generalize convolutional neural networks (CNNs) to graph-structured data from the perspective of spectral theory based on prior works (Bruna et al. 2013; Defferrard, Bresson, and Vandergheynst 2016). GCNs naturally

integrate the connectivity patterns and feature attributes of graph-structured data and it has been demonstrated that GCNs and their variants (Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2017; Dai et al. 2018; Chen and Zhu 2017) significantly outperform traditional multi-layer perceptron (MLP) models and traditional graph embedding approaches (Tang et al. 2015; Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016).

Nevertheless, it is well known that deep neural networks heavily depend on a large amount of labeled data. The requirement of large-scale data might not be met in many real scenarios for graphs with sparse labeled nodes. GCNs and their variants are mainly established on semi-supervised setting where the graph usually has relative plenty of labeled data. However, to the best of our knowledge, there is hardly any work about graphs focusing on weakly supervised setting (Zhou 2017), especially learning a classification model with few examples from each class. In addition, the GCNs are usually with shallow architectures due to its intrinsic limitation (Li, Han, and Wu 2018), thereby restricting the efficient propagation of label signals. To address this issue, (Li, Han, and Wu 2018) proposed Co-Training and Self-Training to enlarge training dataset in a boosting-like way. Although these methods can partially improve the performance of GCNs with few labeled data, it is difficult to pick single one consistently efficient algorithm in real applications since these proposed methods (Li, Han, and Wu 2018) perform inconsistently across distinct training sizes.

On the other hand, a recent surge of interest has focused on the self-supervised learning, a popular form of unsupervised learning, which uses pretext tasks to replace the labels annotated by humans by "pseudo-label" directly computed from the raw input data. On the basis of the analysis above, there are mainly two issues worthy to explore further. Firstly, since it is hard to change the innate shallow architectures of GCNs, how to design a consistently efficient training algorithm based on GCNs to improve its generalization performance on graphs with few labeled nodes? Secondly, how to leverage the advantage of self-supervised learning approaches based on a large amount of unlabeled data, to refine the performance of proposed training algorithm?

In this paper, we firstly analyze the Symmetric Laplacian

Smoothing (Li, Han, and Wu 2018) of GCNs and show that this intrinsic property determines the shallow architectures of GCNs, thus restricting its generalization performance on only few labeled data due to the inefficient propagation of label information. Then we show the layer effect of GCNs on graph with few labeled nodes: to maintain the best generalization, it requires more layers for GCNs with fewer labeled data in order to propagate the weak label signals more broadly. Further, to overcome the inefficient propagation of label information with few labels for shallow architectures of GCNs, we firstly propose a more general training algorithm of GCNs based on Self-Training (Li, Han, and Wu 2018), called *Multi-Stage Training Framework*. Furthermore, we apply DeepCluster (Caron et al. 2018), a popular method of self-supervised learning, on the graph embedding process of GCNs and design a novel aligning mechanism on clusters to construct pseudo-labels in classification for each unlabeled data in the embedding space. Next we incorporate DeepCluster approach and the aligning mechanism into the Multi-Stage Training Framework in an elegant way and formally propose *Multi-Stage Self-Supervised (M3S) Training Algorithm*. Extensive experiments demonstrate that our M3S approach are superior to other state-of-the-art approaches across all the considered graph learning tasks with limited number of labeled nodes. In summary, the contributions of the paper are listed below:

- We first probe the existence of Layer Effect of GCNs on graphs with few labeled nodes, revealing that GCNs requires more layers to maintain the performance with lower label rate.

- We propose an efficient training algorithm, called M3S, combining the Multi-Stage Training Framework and DeepCluster approach. It exhibits state-of-the-art performance on graphs with low label rates.

- Our M3S Training Algorithm in fact can provide a more general framework that leverages self-supervised learning approaches to improve multi-stage training framework to design efficient algorithms on learning tasks with only few labeled data.

Our Approach

Before introducing our M3S training algorithm, we will firstly elaborate the issue of inefficient propagation of information from limited labeled data due to the essence of symmetric laplacian smoothing of GCNs, which forms the motivation of our work. Then a multi-stage training framework and DeepCluster approach are proposed, respectively, composing the basic components of our M3S algorithm. Finally, we will formally provide multi-stage self-supervised (M3S) training algorithm in detail, a novel and efficient training method of GCNs focusing on graphs with few labeled nodes.

Symmetric Laplacian Smoothing of Graph Convolutional Networks

In the GCNs model (Kipf and Welling 2016) of semi-supervised classification, the graph embedding Z of nodes with two convolutional layers is formulated as:

$$Z = \text{softmax}(\hat{A}\,\text{ReLU}(\hat{A}XW^{(0)})W^{(1)}), \qquad (1)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I$ and $\tilde{D}$ is the degree matrix of $\tilde{A}$. X and A denote the feature and the adjacent matrix, respectively. $W^{(0)}$ is the input-to-hidden weight matrix and $W^{(1)}$ is the hidden-to-output weight matrix.

Related work (Li, Han, and Wu 2018) pointed out the reason why the GCNs work lies in the Symmetric Laplacian Smoothing of this spectral convolutional type, which is the key for the huge performance gain. We simplify it as follows:

$$\mathbf{z}_i = \sum_j \frac{\tilde{a}_{ij}}{\sqrt{\tilde{d}_i}\sqrt{\tilde{d}_j}}\mathbf{x}_j \quad (\text{for } 1 \le i \le n), \qquad (2)$$

where n is the size of nodes and $\mathbf{z_i}$ is the first-layer embedding of node i from input features $\mathbf{x}$. Its corresponding matrix formulation is as follows:

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X, \qquad (3)$$

where Z is the one-layer embedding matrix of feature matrix X. In addition, (Li, Han, and Wu 2018) showed that by repeatedly applying Laplacian smoothing many times, the embedding of vertices will finally converge to the proportional to the square root of the vertex degree, thus restricting the enlargement of convolutional layers.

In this case, a shallow GCN cannot sufficiently propagate the label information to the entire graph with only a few labels, yielding the unsatisfying performance of GCNs on graphs with few labeled nodes. To tackle this deficit of GCNs, we propose an effective training algorithm based on GCNs especially focusing on graphs with only a small number of labels, dispensing with the inconsistent performance of four algorithms proposed in (Li, Han, and Wu 2018).

On the other hand, as shown in Figure 2, the requirement of number of graph convolutional layers for the best performance differs for the different label rates. Concretely speaking, the lower label rate of a graph has, the more graph convolutional layers are required for the purpose of more efficient propagation of label information.

Multi-Stage Training Framework

Inspired by the Self-Training algorithm proposed by (Li, Han, and Wu 2018), working by adding the most confident predictions of each class to the label set, we propose a more general Multi-Stage Training Framework described in Algorithm 1.

In contrast with original Self-Training that explores the most confident nodes and adds them with predicted virtual labels only once, Multi-Stage Training Algorithm executes this process K times. On graphs with limited labels, this algorithm framework repeatedly adds more confident labeled data and facilitates the propagation of label information, resulting in the better performance compared with original approaches.

Nevertheless, the core of Multi-Stage Training Framework lies in the accuracy of selected nodes with virtual labels based on the confidence and thus it is natural to incorporate

self-checking mechanism that can guarantee the precision of chosen labeled data.

Algorithm 1 Multi-Stage Training Algorithm

Input: Features matrix X, adjacent matrix A, labeled and unlabeled set L_0, U_0, graph convolution network f_θ
Output: Graph Embedding $Z = f_\theta(X, A)$

1: Train a fixed number of epoches on the initial labeled and unlabeled set L_0, U_0
2: **for** each stage k **do**
3:     Sort vertices on confidence in unlabeled set U_{k-1}.
4:     **for** each class j **do**
5:         Find the top t vertices in $Z_{i,j}$.
6:         Add them to labeled set L_{k-1} with virtual labels j.
7:         Delete them from unlabeled set U_{k-1}.
8:     **end for**
9:     Train a fixed number of epoches on the new labeled and unlabeled set L_k, U_k
10: **end for**
11: **return** Accuracy based on the final Z.

DeepCluster

Recently, self-supervised learning (Doersch, Gupta, and E-fros 2015), a popular form of unsupervised learning, shows its power in the field of computer vision, which utilizes pretext tasks to replace the labels annotated by human by "pseudo-labels". A neat and effective approach of self-supervised learning is DeepCluster (Caron et al. 2018) that takes a set of embedding vectors produced by ConvNet F as input and groups them into k distinct clusters based on a geometric criterion.

More concretely, DeepCluster jointly learns a $d \times k$ centroid matrix C and the cluster assignment y_n of each data point n such as image, by solving the following problem:

$$\min_{C \in \mathcal{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^k} \|F(x_n) - C y_n\|_2^2 \tag{4}$$
$$s.t. \quad y_n^T 1_k = 1$$

Solving this problem provides a set of optimal assignments $(y_n^*)_{n \le N}$ and a centroid matrix C^*. These assignments are then used as pseudo-labels. In particular, Deep-Cluster alternates between clustering the embedding vectors produced from ConvNet into pseudo-labels and updating parameters of the ConvNet by predicting these pseudo-labels.

For the node classification task in a graph, the representation process can also be viewed as graph embedding (Zhou et al. 2018), allowing the DeepCluster as well. Thus, we harness the innate property of graph embedding in GCN-s and execute k-means on the embedding vectors to cluster all nodes into distinct categories based on embedding distance. Next, an aligning mechanism is introduced to classify the nodes in each cluster to the nearest class in classification on the embedding space. Finally, the obtained pseudo-labels are leveraged to construct the self-checking mechanism of Multi-Stage Self-Supervised Algorithm as shown in Figure 1.

Aligning Mechanism The target of aligning mechanism is to transform the categories in clustering to the classes in classification based on the embedding distance. For each cluster l in unlabeled data after k-means, the computation of aligning mechanism is:

$$c^{(l)} = \arg\min_m \|v_l - \mu_m\|^2, \tag{5}$$

where μ_m denotes centroids of class m in labeled data, v_l denotes the centroid of cluster l in unlabeled data and $c^{(l)}$ represents the aligned class that has the closest distance from v_l among all centroids of class in the original labeled data.
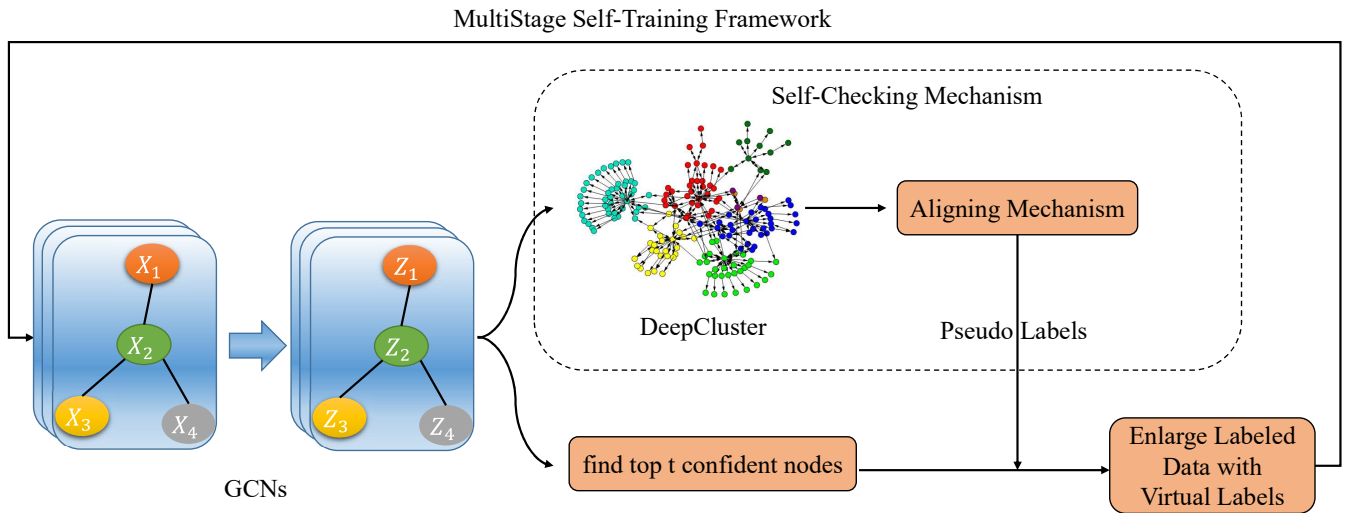

Figure 1: Flow chart of Multi-Stage Self-Supervised (M3S) Training Algorithm.

Through the aligning mechanism, we are capable of classifying nodes of each cluster to a specific class in classification and then construct pseudo-labels for all unlabeled nodes according to their embedding distance.

Extension In fact, DeepCluster is a more general and economical form of constructing self-checking mechanism via embedding distance. The naive self-checking way is to compare the distance of each unlabeled node to centroids of classes in labeled data since distance between each unlabeled data and training centroids is a more precise measure than the class centriods of unlabeled data. However, when the number of clusters is equivalent to the amount of all unlabeled nodes, our self-checking mechanism via DeepCluster is the same as the naive way. Considering the expensive computation of the naive self-checking, DeepCluster performs more efficiently and flexibly in the selection of number of clusters.

M3S Training Algorithm

In this section, we will formally present our Multi-Stage Self-Supservised (M3S) Training Algorithm, a novel training method on GCN aiming at addressing the inefficient propagation of label information on graphs with few labeled nodes. The flow chart of our approach is illustrated in Figure 1.

The crucial part of M3S Training Algorithm compared with Multi-Stage Training is additionally utilizing the information of embedding distance to check the accuracy of selected nodes with virtual labels from Self-Training based on the confidence. Specifically speaking, M3S Training Algorithm elegantly combines DeepCluster self-checking mechanism with Multi-Stage Training Framework to choose nodes with more precise virtual labels in an efficient way. We provide a detailed description of M3S approach in Algorithm 2.

For M3S Training Algorithm, firstly we train a GCN model on an initial dataset to obtain meaningful embedding vectors. Then we perform DeepCluster on the embedding vectors of all nodes to acquire their clustering labels. Furthermore, we align their labels of each cluster based on the embedding distance to attain the pseudo-label of each unlabeled node. In the following Self-Training process, for the selected top confident nodes of each class, we perform self-checking based on pseudo-labels to guarantee they belong to the same class in the embedding space, then add the filtered nodes to the labeled set and execute a new stage Self-Training.

Avoiding Trivial Solutions It should be noted that the categorically balanced labeled set plays an important role on graphs with low label rate. In addition, DeepCluster tends to be caught in trivial solutions that actually exist in various methods that jointly learns a discriminative classifier and the labels (Caron et al. 2018). Highly unbalanced data of per class is a typical trivial solution of DeepCluster, which hinders the generalization performance with few supervised signals. In this paper we provide a simple and elegant solution by enlarging the number of clusters in K-means. For the one hand, setting more clusters allows higher probability of

---

Algorithm 2 M3S Training Algorithm

Input: Features Matrix X, adjacent matrix A, labeled and unlabeled set L_0, U_0, graph convolution network f_θ.
Output: Graph Embedding $Z = f_\theta(X, A)$
1: Train a fixed number of epoches on the initial labeled and unlabeled sets L_0, U_0.
2: **for** each stage k **do**
3:    **% Step 1: Deep Clustering**
4:    Execute K-means based on embedding Z of all data and obtain pseudo labels of each data point for clustering.
5:    **% Step 2: Aligning Mechanism**
6:    Compute centroids μ_m of each class m in labeled data.
7:    Compute centroids v_l of each cluster l in unlabeled data.
8:    **for** each cluster l of unlabeled set **do**
9:       Align label of l_{th} cluster on the embedding space.
$$c^{(l)} = \arg\min_m \|v_l - \mu_m\|^2$$
10:       Set unlabeled data in l_{th} cluster with pseudo label $c^{(l)}$.
11:    **end for**
12:    **% Step 3: Self-Training**
13:    Sort vertices according to the confidence in unlabeled set U_{k-1}.
14:    **for** each class j **do**
15:       Find the top t vertices in $Z_{i,j}$.
16:       **for** each vertice of selected t vertices **do**
17:          **if** pseudo label c of the vertice equals j **then**
18:             Add it to labeled set L_{k-1} with virtual label j.
19:             Delete it from unlabeled set U_{k-1}.
20:          **end if**
21:       **end for**
22:    **end for**
23:    Train a fixed number of epoches on the new labeled and unlabeled set L_k, U_k
24: **end for**
25: **return** Accuracy based on the final Z.

---

being evenly classified to all categories. For the other hand, it contributes to more precise computation in embedding distance from the perspective of extension of DeepCluster self-checking mechanism. These are dicussed in the experimental part.

Experiments

In this section we conduct extensive experiments to demonstrate the effectiveness of our proposed M3S Algorithm on graphs with few labeled nodes. For the graph dataset, we select the three commonly used citation networks: CiteSeer, Cora and PubMed (Sen et al. 2008). Dateset statistics are summarized in Table 1.

As for the baselines, we opt the Label Propagation (LP) (Wu et al. 2012) using ParWalks; Graph Convolutional Net-
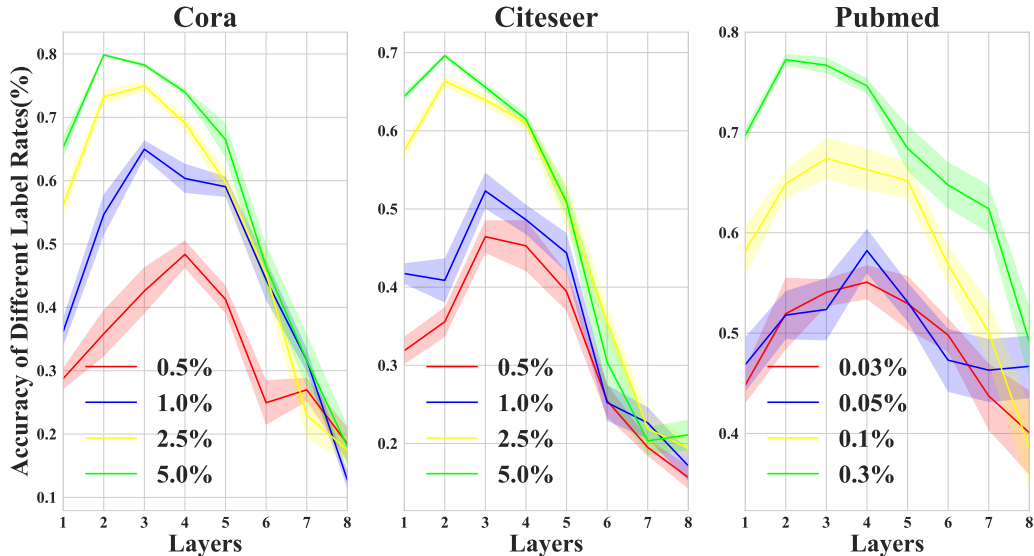
Figure 2: The change of accuracy for GCNs model with different layers under different label rates.

works (GCNs) (Kipf and Welling 2016); Self-Training, Co-Training, Union and Intersection (Li, Han, and Wu 2018) all based on the confidence of prediction. On graphs with low label rates, we compare both our Multi-Stage Training Framework and M3S Algorithm with other state-of-the-art approaches by changing the label rate for each dataset. We report the mean accuracy of 10 runs in all result tables to make fair comparison. Our implementationm, including the splitting of train and test datasets, adapts from original version in (Li, Han, and Wu 2018).

Dateset	Nodes	Edges	Classes	Features	Label Rate
CiteSeer	3327	4732	6	3703	3.6%
Cora	2708	5429	7	1433	5.2%
PubMed	19717	44338	3	500	0.3%

Table 1: Dateset statistics

Layer Effect on Graphs with Few Labeled Nodes

Before comparing our algorithms with other methods, we point out the layer effect of GCNs for different label rates: to maintain the best performance, a GCN model in semi-supervised task with a lower label rate requires more graph convolutional layers.

Figure 2 presents some empirical evidence to demonstrate the layer effect on graphs with few labels. We test the performance of GCNs with different layers in distinct label rates in Figure 2 and it is apparent to note that the number of layer under the best performance exhibits a descending trend as the label rate increases.

The existence of layer effect demonstrates the urge of propagation of label information by stacking more convolutional layers. In the original GCNs (Kipf and Welling 2016),

the authors argued to apply two graph convolutional layers for standard node classification tasks. However, due to the existence of Layer Effect, we are expected to choose proper number of layers especially on graphs with low label rates. In the following experiments, we all choose the best number of layer to compare the best performance for each method.

Performance of Multi-Stage Training Framework

To gain a better understanding of the advantage of Multi-Stage Training Framework, we make an extensive comparison between Multi-Stage Framework of different stages with the Self-Training approach under different label rates.

From Figure 3, it is easy to observe that all self-training methods outperform the original GCNs with a large margin, especially when the graph has low label rate, which usually happens in real applications. In addition, Multi-Stage Training is superior to traditional Self-Training especially when there are fewer labeled nodes and more stages are inclined to bring more improvement. Nevertheless, the discrepancy between the Multi-Stage Training algorithm and Self-Training algorithm narrows down as the label rate increases. Moreover, the improvement of all self-training methods over GCNs diminishes as well with the increasing of label rate. As for the reason, we argue that with the enlargement of labeled nodes, the accuracy of the learned GCN model also increases, while the accuracy of explored nodes via self-training tends to approach the accuracy of current GCN, resulting in the diminishment of improvement. However, the limited precision of selected nodes only based on the confidence of prediction is just what M3S Training Algorithm is devoted to improve.

Performance of M3S Training Algorithm

In this section, we conduct experiments by comparing Multi-Stage Self-Training Algorithm and M3S Training Algorithm
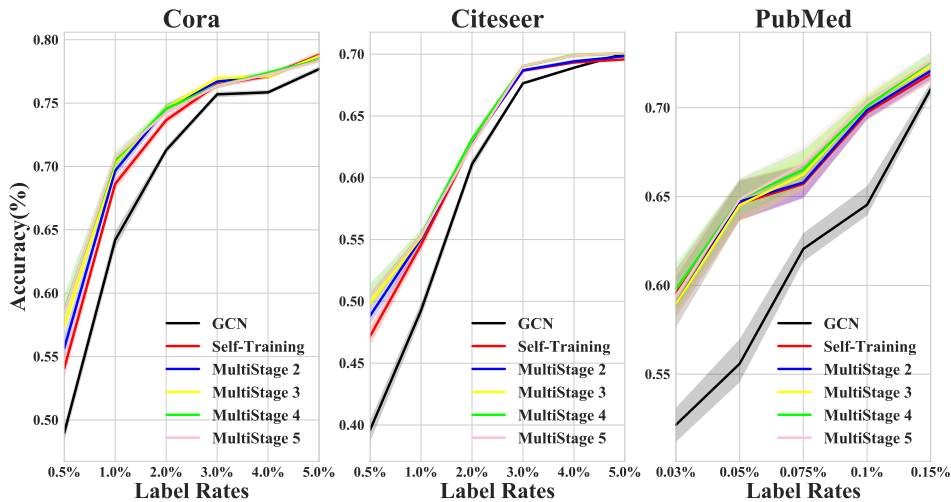
Figure 3: Multi-Stage Training vs Self-training.

with other state-of-the-art approaches under different label rates across the three datasets.

Experimental Setup All the results are the mean accuracy of 10 runs and the number of clusters in DeepCluster is fixed 200 for all datasets to avoid trivial solutions. We select the best number of layers for different label rates. In particular, the best layer in Cora and CiteSeer is 4,3,3,2,2 and 3,3,3,2,2 respectively for 0.5%,1%,2%,3%,4% label rates and fixed 4 for 0.03%,0.05%,0.1% label rates on PubMed. The number of epochs of each stage in Multi-Stage Training Framework, M3S and other approaches is set as 200. For all methods involved in GCNs, we use the same hyper-parameters as in (Kipf and Welling 2016): learning rate of 0.01, 0.5 dropout rate, 5×10^{-4} L_2 regularization weight, and 16 hidden units without validation set for fair comparison (Li, Han, and Wu 2018). For the option of K stages, we view it as a hyper-parameter. For CiteSeer dataset we fix $K = 3$ and for PubMed dataset we fix $K = 4$, in which the result of our proposed algorithms have already outperformed other approaches easily. For Cora dataset we choose K as 5,4,4,2,2 as the training size increases, since higher label rate usually matches with a smaller K.

Results shown in Tables 2, 3 and 4 verify the effectiveness of our M3S Training Algorithm, consistently outperforming other state-of-the-art approaches to a large margin on a wide range of label rates across the three datasets. More specifically, we make four observations from the results:

- It is apparent to note that the performance of GCN significantly declines when the labeled data is scarce due to the inefficient propagation of label information. For instance, on Cora and PubMed datasets, the performance of GCN is even inferior to Label Propagation (LP) when the training size is relative small.

CiteSeer Dataset					
Label Rate	0.5%	1%	2%	3%	4%
LP	37.7	41.6	41.9	44.4	44.8
GCN	44.8	54.7	61.2	67.0	69.0
Co-training	42.0	50.0	58.3	64.7	65.3
Self-training	51.4	57.1	64.1	67.8	68.8
Union	48.5	52.6	61.8	66.4	66.7
Intersection	51.3	61.1	63.0	69.5	70.0
MultiStage	53.0	57.8	63.8	68.0	69.0
M3S	**56.1**	**62.1**	**66.4**	**70.3**	**70.5**

Table 3: Classification Accuracy on CiteSeer.

Cora Dataset					
Label Rate	0.5%	1%	2%	3%	4%
LP	57.6	61.0	63.5	64.3	65.7
GCN	50.6	58.4	70.0	75.7	76.5
Co-training	53.9	57.0	69.7	74.8	75.6
Self-training	56.8	60.4	71.7	76.8	77.7
Union	55.3	60.0	71.7	77.0	77.5
Intersection	50.6	60.4	70.0	74.6	76.0
MultiStage	61.1	63.7	74.4	76.1	77.2
M3S	**61.5**	**67.2**	**75.6**	**77.8**	**78.0**

Table 2: Classification Accuracy on Cora.

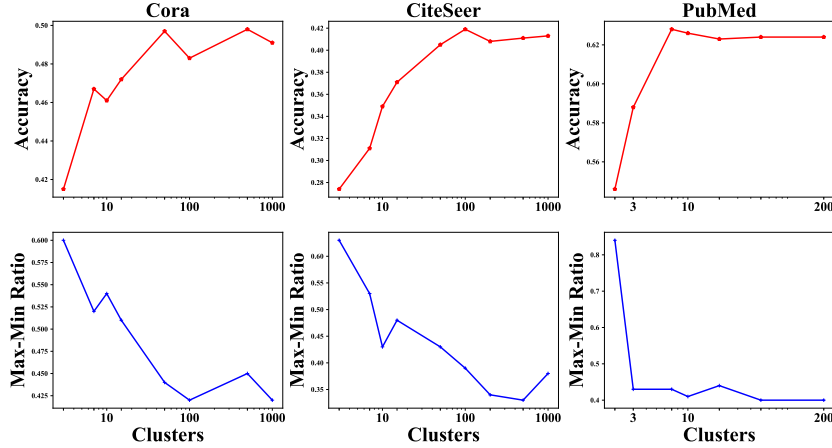| PubMed Dataset			
Label Rate	0.03%	0.05%	0.1%
LP	58.3	61.3	63.8
GCN	51.1	58.0	67.5
Co-training	55.5	61.6	67.8
Self-training	56.3	63.6	70.0
Union	57.2	64.3	70.0
Intersection	55.0	58.2	67.0
MultiStage	57.4	64.3	70.2
M3S	**59.2**	**64.4**	**70.6**

Table 4: Classification Accuracy on PubMed.

Figure 4: Relation between Accuracy and Max-Min Ratio with the increasing of Clusters K. All values are the mean accuracy or max-min ratio of 10 runs.

- Previous state-of-the-art algorithms, namely Co-training, Self-training, Union and Intersection exhibit inconsistent performance compared with GCNs, thus it is hard to employ one single algorithm from them in real scenarios.
- Multi-Stage Training Framework tends to be superior to Self-Training especially on fewer labeled data, demonstrating the effectiveness of this framework on graphs with few labeled nodes.
- M3S Training Algorithm leverages both the advantage of Multi-Stage Training Framework and self-checking mechanism constructed by DeepCluster, consistently outperforming other state-of-the-art approaches on all label rates. Additionally, it turns out that the lower label rate the graph has, the larger improvement of M3S Training Algorithm can produce, perfectly adapting on graphs with few labeled nodes.

Sensitivity Analysis of Number of Clusters Sensitivity analysis of number of clusters is regarded as the extensive discussion of our M3S Training Algorithm, where we present the influence of number of clusters in DeepCluster on the balance of each class and the final performance of GCN. We leverage "Max-Min Ratio" to measure the balance level of each class, which is computed by the subtraction between max ratio and min ratio of categories of unlabeled data after the aligning mechanism, and the lower "Max-Min Ratio" represents the higher balance level of categories. We choose two labeled nodes of each class across three datasets. As shown in Figure 4 where each column presents the change of a specific dataset, with the increasing of number of clusters, categories tend to be more balanced until the number of clusters is large enough, facilitating the final performance of M3S Training Algorithm. These results empirically demonstrate that more clusters are beneficial to avoid trivial solutions in DeepCluster, thus enhancing the performance of our method.

Discussions

Although in this work we employ only one kind of self-supervised approach on the graph learning task, the introduction of self-checking mechanism constructed by Deep-Cluster in fact provides a more general framework on weakly supervised signals for a wide range of data types. On the one hand, it is worthy of exploring the avenue to utilize the pseudo-labels produced by self-supervised learning more efficiently on few supervised labels, for instance, designing new aligning mechanism or applying better self-supervised learning approach. On the other hand, how to extend similar algorithm combined with self-supervised learning methods to other machine learning task such as image classification and sentence classification, requires more endeavours in the future.

Conclusion

In this paper, we firstly clarify the Layer Effect of GCNs on graphs with few labeled nodes, demonstrating that it is expected to stack more layers to facilitate the propagation of label information with lower label rate. Then we propose Multi-Stage Training Algorithm Framework on the basis of Self-Training, adding confident data with virtual labels to the labeled set to enlarge the training set. In addition, we apply DeepCluster on the graph embedding process of GCNs and design a novel aligning mechanism to construct self-checking mechanism to improve MultiStage Training Framework. Our final proposed approach, M3S Training Algorithm, outperforms other state-of-the-art methods with different label rates across all the considered graphs with few labeled nodes. Overall, M3S Training Algorithm is a novel and efficient algorithm focusing on graphs with few labeled nodes.

References

Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J.; et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, 4502–4510.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Computer Vision–ECCV 2018*. Springer. 139–156.

Chen, J., and Zhu, J. 2017. Stochastic training of graph convolutional networks. *International Conference on Machine Learning,ICML 2018*.

Dai, H.; Kozareva, Z.; Dai, B.; Smola, A.; and Song, L. 2018. Learning steady-states of iterative algorithms over graphs. In *International Conference on Machine Learning,ICML 2018*, 1114–1122.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems, NeurIPS 2016*, 3844–3852.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 1422–1430.

Fortunato, S. 2010. Community detection in graphs. *Physics reports* 486(3-5):75–174.

Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, 6530–6539.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

Hamaguchi, T.; Oiwa, H.; Shimbo, M.; and Matsumoto, Y. 2017. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. *arXiv preprint arXiv:1706.05674*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations, ICLR 2017*.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *Association for the Advancement of Artificial Intelligence, AAAI 2019*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Sanchez-Gonzalez, A.; Heess, N.; Springenberg, J. T.; Merel, J.; Riedmiller, M.; Hadsell, R.; and Battaglia, P. 2018. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. International World Wide Web Conferences Steering Committee.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *International Conference on Learning Representations, ICLR 2018* 1(2).

Wu, X.-M.; Li, Z.; So, A. M.; Wright, J.; and Chang, S.-F. 2012. Learning with partially absorbing random walks. In *Advances in Neural Information Processing Systems*, 3077–3085.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

Zhou, Z.-H. 2017. A brief introduction to weakly supervised learning. *National Science Review* 5(1):44–53.

Zhu, J.; Song, J.; and Chen, B. 2016. Max-margin nonparametric latent feature models for link prediction. *arXiv preprint arXiv:1602.07428*.