Active Learning for Graph Neural Networks via Node Feature Propagation

Yuexin Wu*, **Yichong Xu***, **Aarti Singh, Yiming Yang, Artur Dubrawski**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{yuexinw,yichongx,aarti,yiming,awd}@cs.cmu.edu

Abstract

Graph Neural Networks (GNNs) for prediction tasks like node classification or edge prediction have received increasing attention in recent machine learning from graphically structured data. However, a large quantity of node labels is difficult to obtain, which significantly limit the true success of GNNs. Although active learning has been widely studied for addressing label-sparse issues with other data types like text, images, etc., how to make it effective over graphs is an open question for research. In this paper, we present an investigation of active learning with GNNs for node classification tasks. Specifically, we propose a new method, which uses node feature propagation followed by K-Means clustering of the nodes for instance selection in active learning. With a theoretical bound analysis, we justify the design choice of our approach. In our experiments on four benchmark dataset, the proposed method outperforms other representative baseline methods consistently and significantly.

1 Introduction

Graph Neural Networks (GNN) [15, 19, 9, 20] have been widely applied in many supervised and semi-supervised learning scenarios such as node classifications, edge predictions and graph classifications over the past few years. Though GNN frameworks are effective at fusing both the feature representations of nodes and the connectivity information, people are longing for enhancing the learning efficiency of such frameworks using limited annotated nodes. This property is in constant need as the budget for labeling is usually far less than the total number of nodes. For example, in biological problems where a graph represents the chemical structure [8, 11] of a certain drug assembled through atoms, it is not easy to obtain a detailed analysis of the function for each atom since getting expert labeling advice is very expensive. On the other hand, people can carefully design a small "seeding pool" so that by selecting "representative" nodes or atoms as the training set, a GNN can be trained to get an automatic estimation of the functions for all the remaining unlabeled ones.

Active Learning (AL) [18, 2], following this lead, provides solutions that select "informative" examples as the initial training set. The most common approach in AL is uncertainty sampling [18] where the nodes that the current model is least certain about along with the training phase are selected. Another major stream of AL approaches is the clustering-based selection strategy. Coreset [17], as one representative method in this category, generates several anchor points so that each node's feature representation is within a close proximity to one of the anchors. Nodes that are closest to the anchors are selected as the representative initial training set.

However, our experiments (see Figure 1) show that these existing methods perform similarly to random sampling in most cases when applied to the graph domain. We conjecture that this is because none of these methods expressly consider the downstream classification algorithms (i.e.

*Equal contribution.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

GNN frameworks) which uses the graph structure to achieve a good performance. As traditional AL settings do not include such graphs, the selected samples therefore might not be representative enough when combined with GNN frameworks. In this paper we propose a solution to GNN frameworks in the graph learning setting, to address this issue with the explicit consideration of GNN structures.

In this work, we present a novel lightweight AL method specifically designed for the Graph Convolution Network (GCN) [15], one of the most widely-used GNN structures. Our method performs the node feature propagation succeeded by the K-Means clustering to select representative nodes so that the information of the graph structure and the node features are fused in a similar way as the GCN does. We also compare our methods systematically with state-of-the-art AL methods. The contributions of our work are mainly three-fold:

- We propose a new AL method, tailored for GCNs, for the node classification task.
- We perform a theoretical analysis for our method and study the relation between its classification loss and the geometry of the propagated node representations.
- We compare our method with several AL methods and obtain the best performance over all benchmark datasets.

2 Preliminaries

In this section, we describe a formal definition similar to [17] for the problem of inductive active learning under the node classification setting and introduce a uniform set of notations for the rest of the paper.

We are given a large graph $G = (V, E)$, where each node $v \in V$ is associated with a feature vector $x_v \in \mathcal{X} \subseteq \mathbb{R}^d$, and a label $y_v \in \mathcal{Y} = \{1, 2, ..., C\}$. Let $V = \{1, 2, ..., n\}$, we denote the input features as a matrix $X \in \mathbb{R}^{n \times d}$, where each row represents a node, and the labels as a vector $Y = (y_1, ..., y_n)$. We also consider a loss function $l(\mathcal{M}|G, X, Y)$ that computes the loss over the inputs (G, X, Y) for a model $\mathcal{M}$ that maps G, X to a prediction vector $\hat{Y} \in \mathcal{Y}^n$. We focus on $\mathcal{M}$ being Graph Neural Networks and their variants elaborated in detail in the following sections.

An active learning algorithm $\mathcal{A}(\mathcal{M})$ is initially given the graph G and feature matrix X. In step t of operation, it selects a subset $\mathbf{s}^t \subseteq [n] = \{1, 2, ..., n\}$, and obtains y_i for every $i \in \mathbf{s}^t$. We assume y_i is drawn randomly according to a distribution $\mathbb{P}_{y|x_i}$ supported on $\mathcal{Y}$; we use $\eta_c(v) = \Pr[y = c|v]$ to denote the probability that $y = c$ given node v. Then $\mathcal{A}(\mathcal{M})$ uses G, X and y_i for $i \in \mathbf{s}^0 \cup \mathbf{s}^1 \cup \cdots \cup \mathbf{s}^t$ as the training set to train the model $\mathcal{M}$. The trained model $\mathcal{M}$ is denoted as $\mathcal{M}_{\mathcal{A}_t}$. If $\mathcal{M}$ is the same for all active learning strategies, we can slightly abuse the notation $\mathcal{A}_t = \mathcal{M}_{\mathcal{A}_t}$ to emphasize the focus of active learning algorithms. A general goal of active learning is then to minimize the loss under a given budget b:

$$\min_{\mathbf{s}^0 \cup \cdots \cup \mathbf{s}^t} \mathbb{E}[l(\mathcal{A}_t|G, X, Y)] \tag{1}$$

where the randomness is over the random choices of Y and $\mathcal{A}$.

3 Active Learning Strategy & Theoretical Analysis

Traditionally, active learning algorithms choose one instance at a time for labeling, i.e., with $|\mathbf{s}^t| = 1$. However, for modern datasets where the numbers of training instances are very large, it would be extremely costly if we re-train the entire system each time when a new label is obtained. Hence we focus on the one-time "batched" active learning setting [6], which is also called the *optimal experiment design* in the literature [16, 1]. Aiming to select the b most representative nodes as the batch once and for all, our target (1) becomes:

$$\min_{|\mathbf{s}^0| \leq b} \mathbb{E}[l(\mathcal{A}_t|G, X, Y)]. \tag{2}$$

The node selection algorithm is described in Section 3.1, followed by the loss bound analysis in Section 3.2. We also include the comparison with a closely related algorithm (K-Center in Coreset [17]) in Section B.

3.1 Node Selection via Feature Propagation and K-Means Clustering

Our algorithm acts in two steps: 1) computing a distance matrix or function $d_{X,G}$ using the node feature representations X and the graph structure G; 2) applying clustering with b centers over

Algorithm 1 Active Learning based on Clustering

Require: Node representation matrix X, graph structure G and budget b
1: Compute a distance function $d_{X,G}(\cdot,\cdot) : V \times V \to \mathbb{R}$
2: Perform clustering using $d_{X,G}$ with b centers
3: Select $\mathbf{s}$ to be the nodes that are closest (under $d_{X,G}$ measure) to these centers
4: **return** $\mathbf{s}$

this distance matrix, and from each cluster select the node closest to the center of the cluster. The procedure is described in detail in Algorithm 1. Generally speaking, different options for these two steps would yield different performances in the down-stream prediction tasks, which we will analyze theoretically in Section 3.2 and examine empirically in Section 4.1.

Note that our distance matrix depends on both the node features in X and the graph structure in G, while traditional active learning strategies only use X but do not take G into consideration [3, 17]. Differently, we define the pairwise node distance using the L_2 norm of the difference between the corresponding propagated node features: $d_{X,G}(v_i, v_j) = \|(S^K X)_i - (S^K X)_j\|_2$, where $(M)_i$ denotes the i-th row of matrix M and S is the normalized adjacency matrix with added self-loops: $S = (I + D)^{-\frac{1}{2}}(A + I)(I + D)^{-\frac{1}{2}}$ (D is the diagonal degree matrix of adjacency matrix A). Intuitively, the propagated features in our approach act as the surrogates of the GCN hidden representations. For the clustering part, we propose to apply the K-Means clustering to the distance matrix with each row being treated as node representations.

We call our method *FeatProp*, to emphasize the active learning strategy via node feature propagation over the input graph, which is the major difference from other node selection methods (Section 4.1).

3.2 Theoretical Analysis of Classification Loss Bound

Intuitively, representation $S^K X$ resembles the output of a simplified GCN [20] by dropping all activation functions and layer-related parameters in the original structure, which introduces a strong inductive bias and could possibly contribute to the stablized training phase. The following theorem formally shows that using K-Means with propagated features can lead to a low classification loss:

Theorem 1 (informal)**.** *Suppose that the label vector Y is sampled independently from the distribution $y_i \sim \eta(i)$, and the loss function l is bounded by $[-L, L]$. Then under mild assumptions, there exists a constant c_0 such that with probability $1 - \delta$ the expected classification loss of $\mathcal{A}_t$ satisfies*

$$\frac{1}{n} l(\mathcal{A}_t | G, X, Y) \leq \frac{c_0}{n} \sum_{i=1}^{n} \min_{j \in \mathbf{s}^0} \|(S^K X)_i - (S^K X)_j\|_2 + \sqrt{\frac{L \log(1/\delta)}{2n}} \tag{3}$$

To understand Theorem 1, notice that the first term $\sum_{i=1}^{n} \min_{j \in \mathbf{s}^0} \|(S^2 X)_i - (S^2 X)_j\|_2$ is exactly the target loss of K-Means (sum of point-center distances), and the second term $\sqrt{\frac{L \log(1/\delta)}{2n}}$ quickly decays with n, where n is the total number of nodes in graph G. Therefore the classification loss of $\mathcal{A}_t$ on the entire graph G is mostly dependent on the K-Means loss. In practice, we can utilize existing robust initialization algorithms such as K-Means++ to approximate the optimal solution for K-Means clustering.

The assumptions we made in Theorem 1 are pretty standard in the literature, and we illustrate these assumptions in the appendix. While our results share some common characteristics with Sener et al.[17], our proof is more involved in the sense that it relates to the translated features $\|(S^K X)_i - (S^K X)_j\|_2$ instead of the raw features $\|(X)_i - (X)_j\|_2$. In fact, we conjecture that using raw feature clustering selection for GCN will not result in a similar bound as in (3): this is because GCN uses the matrix S to diffuse the raw features across all nodes in V, and the final predictions of node i will also depend on its neighbors as well as the raw feature $(X)_i$. We could see a clearer comparison in practice in Section 4.1.

4 Experiment

We evaluate the node classification performance of our selection method on the Cora, Citeseer, and PubMed network datasets [22]. We further supplement our experiment with an even denser network dataset CoraFull [4] to illustrate the performance differences on a large-scale setting.

We compared our AL strategy (FeatProp) with the following representative strategies: **Random:** Choosing the nodes uniformly from the whole vertex set. **Degree:** Choosing the nodes with the largest degrees. Note that this method does not consider the information of node features. **FeatureClustering:** This method performs a K-Means clustering over the raw features. Nodes that have the closest representations to the centers are selected. This method does not consider the graph structure. **Uncertainty:** Similar to the methods in Joshi et al.[12], we put the nodes with max-entropy and max-margin (i.e. difference between best and second-best scores) into the pool of instances. **Coreset [17]:** This method performs a K-Center clustering over the last hidden representations in the network. If time allows (on Cora and Citeseer), a robust mixture integer programming method as in Sener et al.[17] (dubbed CoresetMIP) is adopted. We also apply their time-efficient approximation version (Coreset-greedy) for all of the datasets. The center nodes are then selected into the pool.

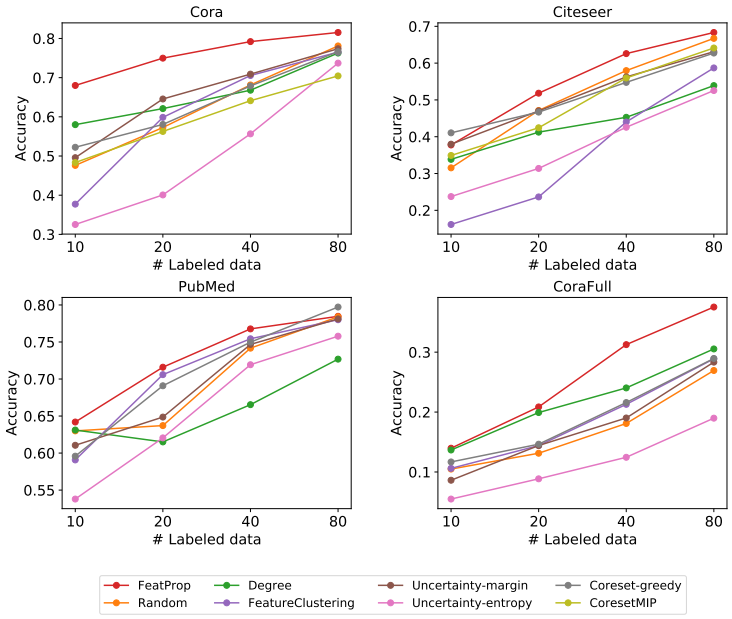

Figure 1: Results of different approaches over benchmark datasets averaged from 5 different runs.

4.1 Experiment Results

Main results. As is shown in Figure 1, our method consistently outperforms all the other baseline methods in all comparing benchmark datasets. It is noticeable that Random could achieve relatively good performances on small datasets (Cora and Citeseer). When the vertex size (PubMed and CoraFull) grows, which may require more insightful guidance from graphs and features, Random will no longer give out good scores. Moreover, it is interesting to find that in CoraFull with the densest linkage structure, Degree, which utilizes the graph information alone could produce a good result and is even better than other active learning baselines. We believe that this is due to the fact that methods (Uncertainty/Coreset) requiring model statuses/hidden representations can be highly-unstable and are more likely to sample biased nodes with an overfitted model resulting from the small number of training points. In this case, strong guidance from the graph structure helps more. Note that Coreset and Uncertainty focus on exactly different selection criteria – either expressiveness of features or the current model confidence. We believe that the observation there is no clear clue for which method is superior is partially due to the fact that datasets of different characteristics favor distinct selection aspects. We include more results in the appendix.

5 Conclusion

We study the active learning problem in the node classification task for Graph Convolution Networks (GCNs). We propose a propagated node feature selection approach (FeatProp) to comply with the specific structure of GCNs and give a theoretical result characterizing the relation between its classification loss and the geometry of the propagated node features. Our empirical experiments also show that FeatProp outperforms the comparing AL methods on most benchmark datasets.

References

[1] Z. Allen-Zhu, Y. Li, A. Singh, and Y. Wang. Near-optimal discrete optimization for experimental design: A regret minimization approach. CoRR, abs/1711.05174, 2017.

[2] Z. Bodó, Z. Minier, and L. Csató. Active learning with clustering. In Active Learning and Experimental Design workshop In conjunction with AISTATS 2010, pages 127–139, 2011.

[3] Z. Bodó, Z. Minier, and L. Csató. Active learning with clustering. In I. Guyon, G. C. Cawley, G. Dror, V. Lemaire, and A. R. Statnikov, editors, Active Learning and Experimental Design workshop, In conjunction with AISTATS 2010, Sardinia, Italy, May 16, 2010, volume 16 of JMLR Proceedings, pages 127–139. JMLR.org, 2011.

[4] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815, 2017.

[5] A. Choromanska, Y. LeCun, and G. B. Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In Conference on Learning Theory, pages 1756–1760, 2015.

[6] G. Contardo, L. Denoyer, and T. Artières. A meta-learning approach to one-step active learning. arXiv preprint arXiv:1706.08334, 2017.

[7] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. arXiv preprint arXiv:1811.03804, 2018.

[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org, 2017.

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

[10] S. Har-Peled. Geometric approximation algorithms. Number 173. American Mathematical Soc., 2011.

[11] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. arXiv preprint arXiv:1802.04364, 2018.

[12] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2372–2379. IEEE, 2009.

[13] K. Kawaguchi. Deep learning without poor local minima. In Advances in neural information processing systems, pages 586–594, 2016.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

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

[16] F. Pukelsheim. Optimal design of experiments. SIAM, 2006.

[17] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489, 2017.

[18] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

[20] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153, 2019.

[21] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. arXiv preprint arXiv:1806.03536, 2018.

[22] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. arXiv preprint arXiv:1603.08861, 2016.

A Proof of Theorem 1

In order to show Theorem 1, we make the following assumptions:
Assumption 1. We assume $\mathcal{A}_t$ has zero training loss on $\mathbf{s}^0$. We also assume that there exists a ground truth GCN $\mathcal{M}^*$ that achieves zero loss on the entire training set. This is a common assumption in literature, and [7] shows that gradient descent provably achieves zero training loss in polynomial time.
Assumption 2. We assume l is Lipschitz with constant λ and bounded in $[-L, L]$.
Assumption 3. We assume that there exists a constant α such that the sum of input weights of every neuron is less than α. Namely, we assume $\sum_i |(W_K)_{i,j}| \leq \alpha$. This assumption is also present in [17]. We note that one can make $\sum_i |(W_K)_{i,j}|$ arbitrarily small without changing the network prediction.
Assumption 4. We assume that ReLU function activates with probability 1/2. This is a common assumption in analyzing the loss surface of neural networks, and is also used in [5, 13, 21].

With these assumptions in place, we are able to prove Theorem 1.

Theorem 1 (restated). Suppose Assumptions 1-4 hold, and the label vector Y is sampled independently from the distribution $y_v \sim \eta(v)$ for every $v \in V$. Then with probability $1 - \delta$ the expected classification loss of $\mathcal{A}_t$ satisfies

$$\frac{1}{n} l(\mathcal{A}_t | G, X, Y) \leq \frac{(\lambda + L)\alpha^K}{n} \sum_{i=1}^{n} \min_{j \in \mathbf{s}^0} \|(S^K X)_i - (S^K X)_j\|_2 + \sqrt{\frac{L \log(1/\delta)}{2n}}.$$

Proof. For simplicity, for any model $\mathcal{M}$ let $(\mathcal{M})_i = (\mathcal{M}(G, X))_i \in \mathbb{R}^C$ be the prediction for node i under input G, X, and $(\mathcal{M})_{i,c}$ be the c-th element of $(\mathcal{M})_i$ (i.e., the prediction for class c). Let $i \in V$ be any node and $j \in \mathbf{s}^0$. We have

$$\begin{aligned}
\mathbb{E}_{y \sim \eta(i)} \left[l((\mathcal{A}_t)_i, y) \right] &= \sum_{c=1}^{C} \Pr[Y_i = c] l((\mathcal{A}_t)_{i,c}, c) \\
&= \sum_{c=1}^{C} \Pr[Y_j = c] l((\mathcal{A}_t)_{i,c}, c) + \sum_{c=1}^{C} \left(\Pr[Y_i = c] - \Pr[Y_j = c] \right) l((\mathcal{A}_t)_{i,c}, c).
\end{aligned} \tag{4}$$

For the first term we have

$$\begin{aligned}
l((\mathcal{A}_t)_{i,c}, c) - l((\mathcal{A}_t)_{j,c}, c) &\leq \lambda |(\mathcal{A}_t)_{i,c} - (\mathcal{A}_t)_{j,c}| \\
&= \lambda \left| \sigma((SX^{(K-1)})_i W_K^c) - \sigma((SX^{(K-1)})_j W_K^c) \right| \\
&\leq \lambda \left| (SX^{(K-1)})_i W_K^c - (SX^{(K-1)})_j W_K^c \right| \\
&\leq \lambda \alpha \left\| (SX^{(K-1)})_i - (SX^{(K-1)})_j \right\| \\
&\leq \lambda \alpha^2 \left\| (S^2 X^{(K-2)})_i - (S^2 X^{(K-2)})_j \right\| \\
&\leq \cdots \leq \lambda \alpha^K \left\| (S^K X)_i - (S^K X)_j \right\|.
\end{aligned}$$

Here $(M)_i$ is the i-th row of matrix M, W_K^c is the c-th column of W_K, σ is the ReLU activation function. The first inequality follows from Lipschitz property of l; the second inequality follows from that ReLU is 1-Lipschitz; the third inequality follows from the definition of α; the next few inequalities follows the same process as above.

Therefore

$$\begin{aligned}\sum_{c=1}^{C}\Pr[Y_j=c]l((\mathcal{A}_t)_{i,c},c) &= \sum_{c=1}^{C}\Pr[Y_j=c]\left[l((\mathcal{A}_t)_{i,c},c)-l((\mathcal{A}_t)_{j,c},c)\right]\\ &\quad+\sum_{c=1}^{C}\Pr[Y_j=c]l((\mathcal{A}_t)_{j,c},c)\\ &=\sum_{c=1}^{C}\Pr[Y_j=c]\left[l((\mathcal{A}_t)_{i,c},c)-l((\mathcal{A}_t)_{j,c},c)\right]\\ &\leq \lambda\alpha^K\left\|(S^KX)_i-(S^KX)_j\right\|\end{aligned}$$

The second equality follows from the fact that $\mathcal{A}_t$ has zero training loss.

Now for the second loss in (4) we use the property that $\mathcal{M}^*$ computes the ground truth:

$$\left(\Pr[Y_i=c]-\Pr[Y_j=c]\right)l((\mathcal{A}_t)_{i,c},c)=\left((\mathcal{M}^*)_{i,c}-(\mathcal{M}^*)_{j,c}\right)l((\mathcal{A}_t)_{i,c},c)$$

We now use the fact that ReLU activates with probability $1/2$, and compute the expectation:

$$\begin{aligned}\mathbb{E}_\sigma\left[\left((\mathcal{M}^*)_{i,c}-(\mathcal{M}^*)_{j,c}\right)l((\mathcal{A}_t)_{i,c},c)\right] &= \mathbb{E}_\sigma\left[\left((\mathcal{M}^*)_{i,c}-(\mathcal{M}^*)_{j,c}\right)\right]l((\mathcal{A}_t)_{i,c}\\ &=\left(\mathbb{E}_\sigma\left[(\mathcal{M}^*)_{i,c}\right]-\mathbb{E}_\sigma\left[(\mathcal{M}^*)_{j,c}\right]\right)l((\mathcal{A}_t)_{i,c}\\ &=\frac{1}{2^K}\left((S^KX)_i-(S^KX)_j\right)W_1W_2\cdots W_K^c l((\mathcal{A}_t)_{i,c}\\ &\leq L\left(\frac{\alpha}{2}\right)^K\|(S^KX)_i-(S^KX)_j\|.\end{aligned}$$

Here $\mathbb{E}_\sigma$ means that we compute the expectation w.r.t randomness in σ (ReLU) in $\mathcal{M}^*$. The last inequality follows from definition of α, and that $l\in[-L,L]$.

Combining the two parts to (4) and we obtain

$$\mathbb{E}_{y\sim\eta(i)}\left[l((\mathcal{A}_t)_i,y)\right]\leq(\lambda+L)\alpha^K\|(S^KX)_i-(S^KX)_j\|.$$

Further let $j=\operatorname{argmin}\|(S^KX)_i-(S^KX)_j\|$, and use Hoeffding's inequality based on $l\in[-L,L]$ gives

$$\frac{1}{n}l(\mathcal{A}_t|G,X,Y)\leq\frac{(\lambda+L)\alpha^K}{n}\sum_{i=1}^{j}\min_j\|(S^2X)_i-(S^2X)_j\|_2+\sqrt{\frac{L\log(1/\delta)}{2n}}$$

□

B Why not K-Center

Besides the propagated features used to compute centers, another important factor of our algorithm is the K-Means clustering. The most related work to our center finding procedure is the K-Center approach in Coreset [17], that tries to minimize the largest point-center distance:

$$\min_{|\mathbf{s}^0|\leq b}\max_i\min_{j\in\mathbf{s}^0}d_{X,G}(v_i,v_j)\tag{5}$$

where the distance function is defined by the dot-product operation between feature representations in the final layer $X^{(K)}$. For the final clustering method (Line 2 in Algorithm 1), we choose to use K-Means instead of K-Center mainly due to two reasons:
1) Under similar assumptions as in Theorem 1, we could see that K-Center is confined to a looser bound in terms of the total loss; that is:

$$\frac{1}{n}l(\mathcal{A}_t|G,X,Y)\leq c_0\max_i\min_{j\in\mathbf{s}^0}\|(S^KX)_i-(S^KX)_j\|_2+\sqrt{\frac{L\log(1/\delta)}{2n}}\tag{6}$$

Theorem 2. *Suppose the set $\mathbf{s}^0$ is a d-cover of the nodes under the distance defined by S^KX, i.e., $\max_{i\in V}\min_{j\in\mathbf{s}^0}\|(S^KX)_i-(S^KX)_j\|_2\leq\delta$. Under the same assumptions and the same constant c_0 as in Theorem 1, with probability $1-\delta$ the expected inductive loss of $\mathcal{A}_t$ satisfies*

$$\frac{1}{n}l(\mathcal{A}_t|G,X,Y)\leq c_0d+\sqrt{\frac{L\log(1/\delta)}{2n}}\tag{7}$$

The first term, proportional to the maximum of point-center distance, is larger than the mean point-center distance counter-part in K-Means. We visualize the difference in Figure 2.
2) As K-Center is NP-Hard [10], in order to get a more accurate solution, people use mixed integer programming (MIP) [17] as the approximated algorithm, which can be much more time-consuming than K-Means based on Expectation Maximization (EM). An alternative solution is to use a greedy selection paradigm specified in Algorithm 2. While being more efficient, the greedy algorithm will have inferior performances to MIP in applications such as image classifications with CNNs [17]. We will elaborate more on these points in the Section 4.1.

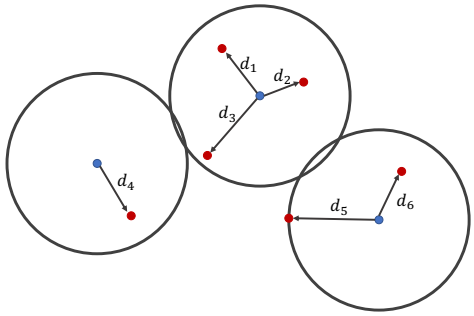

Figure 2: Visualization of Theorem 1. Consider the set of selected points $\mathbf{s}$ and the remaining points in the dataset $[n]\backslash\mathbf{s}$. It suggests from our theorem that the K-Means paradigm gives a bound of $\mathcal{O}(\sum_i d_i/n) + \mathcal{O}(\sqrt{\frac{1}{n}})$ while K-Center (Section B) shows $\mathcal{O}(\max_i d_i) + \mathcal{O}(\sqrt{\frac{1}{n}})$, which is looser than the former one.

Algorithm 2 K-Center-Greedy

Require: Distance function $d_{X,G}(\cdot,\cdot)$, existing pool $\mathbf{s}^0$ and a budget b
1: Initialize $\mathbf{s} = \mathbf{s}^0$ ▷ Warm start, $\mathbf{s}^0$ can be a small random selection
2: **repeat**
3: $u = \arg\max_{i\in[n]\backslash\mathbf{s}} \min_{j\in\mathbf{s}} d_{X,G}(v_i, v_j)$
4: $\mathbf{s} = \mathbf{s} \cup \{u\}$
5: **until** $|\mathbf{s}| = b + |\mathbf{s}^0|$
6: **return** $\mathbf{s}$

C Implementation Details

In our experiments, we start with a small set of nodes (5 nodes) sampled uniformly at random from the dataset as the initial pool. We run all experiments with 5 different random seeds and report the averaged classification accuracy as the metric. We plot the accuracy vs the number of labeled points. For approaches (Uncertainty and Coreset) that require the current status/hidden representations from the classification model, a fully-trained model built from the previous budge pool is returned. For example, if the current budget is 40, the model trained from 20 examples selected by the same AL method is used.

We evaluate the accuracies of the methods over the full set of nodes. The sizes of the budgets are fixed for all benchmark datasets. Specifically, we choose to select $10, 20, 40$ and 80 nodes as the budget sizes. After selecting the nodes, a two-layer GCN [2], with 16 hidden neurons, is trained as the prediction model. We use the Adam [14] optimizer with a learning rate of 0.01 and weight decay of 5×10^{-4}. All the other hyperparameters are kept as in the default setting ($\beta_1 = 0.9, \beta_2 = 0.999$). To guarantee the convergence of the GCN, the model trained after 200 epochs is used to evaluate accuracy on the whole set.

[2]In the past semi-supervised setting of citation networks, a two-layer GCN is the optimal structure for the node classification task [15].

D Addendum to Experiments

Data statistics. Table 1 summarizes the dataset statistics.

Data	# Nodes	# Edges	# Classes	Feature size
Cora	2,708	5,429	7	3,703
Citeseer	3,327	4,732	6	1,433
PubMed	19,717	44,338	3	500
CoraFull	19,793	126,842	70	8,710

Table 1: Dataset statistics of different networks.

	Cora	Citeseer	PubMed	CoraFull
FeatProp	239	622	1,506	13,059
CoresetMIP	12,260	13,257	OOT	OOT
Coreset-greedy	44	46	509	636

Table 2: Comparison of running time of 5 different runs in seconds between our algorithm (FeatProp) and Corset. OOT denotes out-of-time. Note in order to get a more accurate solution, CoresetMIP costs much more time than Coreset-greedy.

Efficiency. We also compare the time expenses between our method and Coreset, which also involves a clustering sub-routine (K-Center, Algorithm 2) in Table 2. It is noticeable that in order to make Coreset more stable, CoresetMIP uses an extreme excess of time comparing to Coreset-greedy in the same setting. An interesting fact we could observe in Figure 1 is that CoresetMIP and Coreset-greedy do not have too much performance difference on Citeseer, and Coreset-greedy is even better than CoresetMIP on Cora. This is quite different from the result in image classification tasks with CNNs [17]. This phenomenon distinguishes the difference between graph node classification with traditional classification problems. We conjecture that this is partially due to the fact that the nodes no longer preserve independent embeddings after the GCN structure, which makes the original analysis of Coreset not applicable.

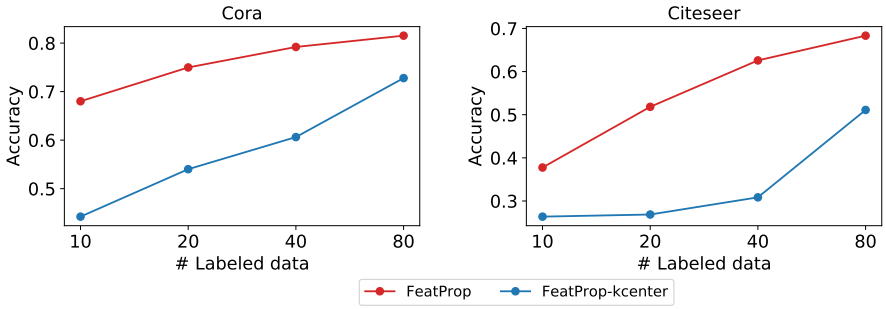

Figure 3: Results of different approaches over benchmark datasets averaged from 5 different runs. FeatProp-kcenter denotes the algorithm replacing the K-Means module with K-Center clustering.

K-Center replacement. It is crucial to select a proper clustering subroutine for FeatProp (Line 2 in Algorithm 1). As is discussed in Section B, we test the differences between the original K-Means choice with a K-Center replacement. We compare these two algorithms in Figure 3. As is demonstrated in the figure, this modified version (FeatProp-kcenter) has a lower accuracy than the original FeatProp approach. This observation is compatible with our analysis in Section B as K-Means comes with a tighter bound than K-Center in terms of the classification loss.

Though FeatProp is tailored for GCNs, we could also test the effectiveness of our algorithm over other GNN frameworks. Specifically, we compare the methods over a Simplified Graph Convolution [20]. We evaluate the performances of different active learning methods on a 2-layer SGC framework. The results can be seen in Figure 4.

Figure 5 shows the results of FeatProp on different GNN frameworks. We see that SGC has a slightly inferior performance to GCN since it drops all the activation functions and in-layer parameters.

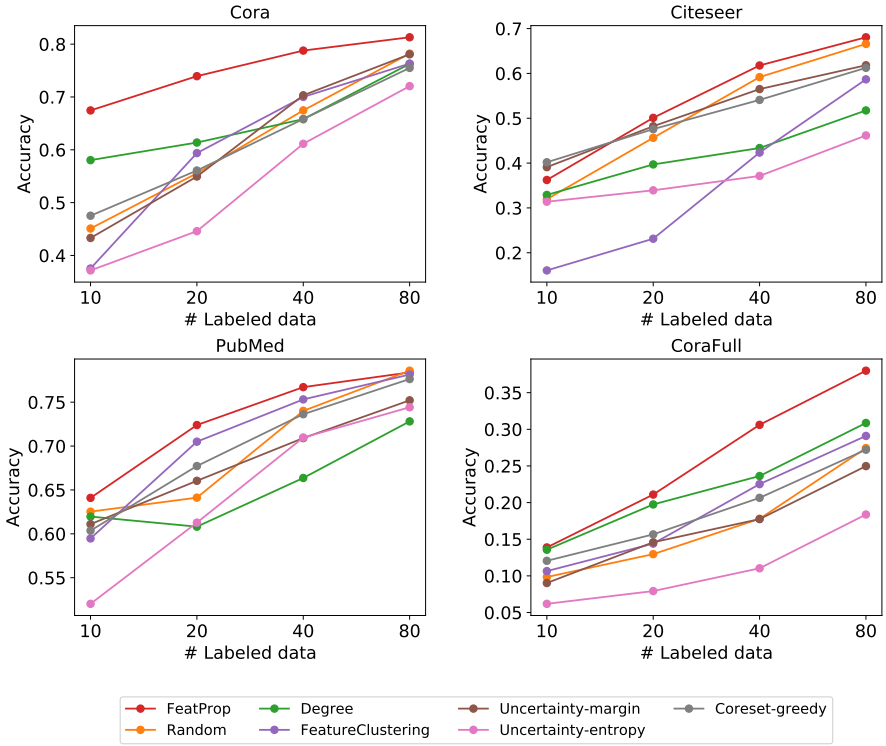

Figure 4: Results of different approaches over benchmark datasets averaged from 5 different runs on an SGC framework.

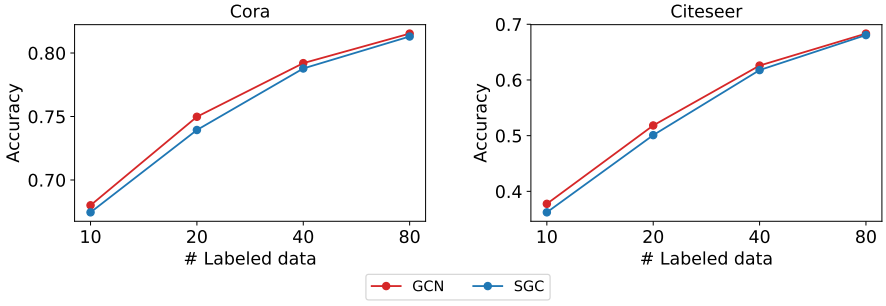

Figure 5: Results of SGC vs GCN over benchmark datasets averaged from 5 different runs by using FeatProp.