Dissimilar Nodes Improve Graph Active Learning

Xiaxuan Gao
gaox8499@gmail.com
University of California, Los Angeles
USA

Zhicheng Ren
franklinnwren@g.ucla.edu
University of California, Los Angeles
USA

Yuxin Wu
yuxinwu98611@g.ucla.edu
University of California, Los Angeles
USA

Yifu Yuan
yiv.yuanyifu@hotmail.com
University of California, Los Angeles
USA

ABSTRACT

Current supervised graph embedding algorithms require a large number of labels in the training data, which is costly to obtain in many practical cases (e.g. the exponentially growing social network graphs). Therefore, developing an efficient method to facilitate graph data labeling has significant impact. Active learning is a protocol that could obtain the most useful labels for training while keeping the total label queries under a certain budget. Information score-based active learning (AL) algorithms is widely used in recent years, due to its negligible computational burden yet significant effectiveness. However, the exploration of information score-based AL algorithms specifically for graph-structured data is still limited. In this paper, we introduce a new concept, dissimilarity, and propose three AL scoring functions based on this idea: feature dissimilarity score (FDS), structure dissimilarity score (SDS), and embedding dissimilarity score (EDS). Our experimental results shows that our proposed method could boost the performance of node classification tasks of Graph Convolutional Networks by about 2.5% when the number of labels is fixed. This implies that our AL scoring functions could select the more valuable nodes to be labeled. Then, we provide an ablation study to show that our methods are generalizable to many other GNN variants. We also examine the effectiveness of our AL scoring functions on heterophilic datasets, and provides some high level explanations why certain scoring functions experience a performance drop.

ACM Reference Format:
Xiaxuan Gao, Zhicheng Ren, Yuxin Wu, and Yifu Yuan. 2018. Dissimilar Nodes Improve Graph Active Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graphs are ubiquitous in the real world such as citation graph in the research area and social network in social media. Nowadays, Graph Neural Networks (GNNs) have been drawing increasing research attention because of its great success in various applications such as node classification [4] and link prediction [8]. However, in many domains such as chemistry and health care, it could be very expensive and time-consuming to collect sufficient amount of labeled data to facilitate the training of GNNS, which significantly limits the performance of GNNs in those domains.

Active learning is a promising strategy to tackle this challenge. The general idea of active learning is to dynamically query the labels of the most informative instances selected from the unlabeled data. In an active learning setup, one is allowed to actively query node labels on the graph given a limited budge constraint. Although active learning has been proven effective on text data in NLP domain and images in CV domain [2], it is still a challenge for the graph mining task because we need to consider the entire graph structure with connections between different entities. This motivates us to study active learning on graphs, e.g. how to effectively find and label the most informative nodes on a graph to reduce the annotation cost of training GNNs.

Traditional active learning query functions for graphs focus on unlabeled nodes only, i.e., how to select the most valuable nodes to get queried within a budget [1]. However, few attention has been put on the nodes which have already been labeled, and how the active learning policy should differ given different sets of unlabeled nodes. It is worth considering the labeled node set and unlabeled node set collectively. Our motivation is that, if a new node sample are dissimilar to all the nodes which have already been labeled, it is likely to contain additional information that could boost the training process of active learning.

In our work, we proposed three novel active learning scores for graphs: feature dissimilarity score (FDS), structure dissimilarity score (SDS), and embedding dissimilarity score (EDS). We conduct extensive experiments and demonstrate that all three scores will improve the performance of learning for graphs by about 2.5% when adding to the other conventionally-used active learning scores. In particular, we observed that a combination of FDS and SDS will yield the best performance in all of our experiments. This aligns with our hypothesis that we need to consider both the individual node features and interactions between different nodes when we do active learning for graphs. Our work provides us with a guidance for future research on this direction. Meanwhile, we also conduct some ablation study where we replace the vanilla GCN backbone with other well-known graph neural network variants such as GAT

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

and SGC. Results show that our methods are generalizable to those GNN variants.

We also examine the effectiveness of our Active Learning (AL) scoring functions on a heterophilic dataset [14], Chameleon, where nodes being connected are likely to belong to different classes. We run experiments for both the traditional AL query scores and our newly proposed dissimilarity scores. We find out most AL query scores experience some performance drop. This suggests a future direction of designing targeted AL query scores for heterophilic graphs.

2 BACKGROUND AND RELATED WORK

2.1 Graph Neural Network

GNN is a family neural networks modeling graph-structured data. They learn node representation by iteratively aggregating neighborhood information of each node. For example, GCN [4] aggregates neighborhood information using a convolution operator, while GAT [11] utilizes an attention mechanism in the aggregation process. Despite their effectiveness, GNNs typically require a set of labeled data for training. By querying the most valuable nodes, we could use less labels to obtain same-level performance as random querying nodes to get annotated.

2.2 General Active Learning Methods

Active learning (AL) is a protocol to train a classifier which is capable of precisely labeling new data based on the limited labeled training samples. AL generally consists of two main components: a query system and a oracle. The main functionality of the query system is to select instances in the training dataset and query the oracle. On the other hand, the oracle predicts the label of the queried instances. During this process, the querying strategy is one of the key things that contribute to a successful AL algorithm. Generally, the querying strategies can be classified into three categories: heterogeneity-based, the performance-based, and the representativeness-based. The heterogeneity-based model is proposed to sample from the space that shares heterogeneous attributes. Apart from that, the performance-based model usually attempts to measure the effect of selected unlabeled sample on the classifier. Besides, the representativeness-based model focuses on designing representativeness criteria to select the most crucial samples from the candidates. One thing to note is that each of these three techniques has certain levels of strengths and weaknesses. Therefore, to design an effective AL algorithm, we need to balance through these strategies.

2.3 AGE

The Active Graph Embedding (AGE) is a framework proposed by [1]. It designs a general active learning query strategy for any semi-supervised graph embedding algorithm. AGE selects the most informative nodes as the training labelled data based on both structural information and learned representiveness. Fig. 1 illustrates the overall architecture of the AGE framework. It takes in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a small portion of labelled nodes as the input. In the next epoch, the Graph Convolutional Network (GCN) module learns on the training dataset for the graph embedding and the node classification. If the labeling budget B is reached at this point, then another epoch of GCN training will be processed. Otherwise, the AL Query Strategy module starts selecting the best candidate in the unlabeled nodes set ($\mathcal{U}$), query it with the oracle, and add it to the labelled nodes set ($\mathcal{L}$). Afterwards, another epoch of GCN will take place. The process is repeated until GCN converges.

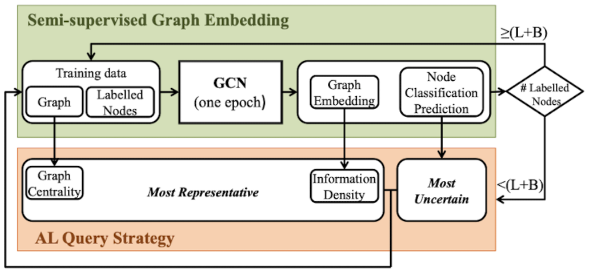

Figure 1: AGE Framework Proposed by Cai et. al. [1]

AGE utilizes three criteria to select the best querying candidates. The first one is a heterogeneous-based metric which uses the entropy information of each node during the learning process. It is calculated according to Eqn. 1:

$$\phi_{\text{entropy}}(v_i) = -\sum_{c=1}^{C} P(Y_{ic} = 1 \mid \mathcal{G}, \mathcal{L}, X) \log P(Y_{ic} = 1 \mid \mathcal{G}, \mathcal{L}, X), \tag{1}$$

where the component $P(Y_{ic} = 1 \mid \mathcal{G}, \mathcal{L}, X)$ denotes the probability of node v_i in the class c.

In addition, AGE proposed two separate representativeness-based metrics: information density (represented as Eqn. 2) and graph centrality (represented as Eqn. 3). The information density measures how representative an candidate node v_i is by using KMeans clustering, the closer a node is to the cluster center, the more representative it is. The second metric measures how important an candidate node v_i is in the perspective of information flow on the entire graph by adopting the PageRank centrality [6]. Their equations are as following:

$$\phi_{\text{density}}(v_i) = \frac{1}{1 + ED\left(Emb_{v_i}, CC_{v_i}\right)} \tag{2}$$

$$\phi_{\text{centrality}}(v_i) = \rho \sum_j A_{ij} \frac{\phi_{\text{centrality}}(v_j)}{\sum_k A_{jk}} + \frac{1-\rho}{N} \tag{3}$$

where the $ED()$ is the euclidean distance function, $Emb()$ is the embedding of the node v_i, CC_{v_i} is the center of cluster which v_i belongs to, and ρ is the damping parameter.

Lastly, to address the scale incompatible issue, AGE assigns each metric a weight parameter such that these weights sum up to 1. Due to the fact that different metrics are more effective at the different periods of the training process, these three weights are time-sensitive and will be assigned a different score as training goes on.

It is notable that the only heterogeneity-based metric in AGE is the entropy score which selects the nodes which the current

GNN model is the most uncertain about. While this is a conventionally adopted strategy, it has a significant drawback: the entropy score is dependent on the current model performance. At the early training state, the models are unlikely to be informative enough to provide a guidance on the uncertainty of the node label prediction. To alleviate this problem, AGE uses a dynamic score function which discourage the use of entropy score at the early training state. However, this means that the protocol cannot select the most informative nodes from the unlabeled node set at the early training state.

2.4 Other Notable Active Learning Model for Graphs

There are some other active learning protocols for GNN-based graph mining models. Hu et. al. [3] propose a performance-based active learning framework based on reinforcement learning. Wu et. al. [13] improve on the KMeans clustering method and yield better accuracy among representativeness-based metrics. Recently, Ma et. al. [5] propose a partition algorithm to enhance the representativeness of the selected nodes. Although Hu et. al.'s work thoroughly exploits the strength of performance-based metrics, the reinforcement learning protocol itself could be a significant computational burden. While both Wu et. al.'s work and Ma et. al.'s work focus on improving representativeness-based metrics only, It is worth to study new categories of metrics and combine them with existing metrics to achieve a more holistic active learning protocol.

3 PROBLEM FORMULATION AND METHODOLOGY

The input of the active graph learning problem includes a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{L}$ is the set of all labeled nodes, along with its adjacency matrix A, its degree matrix D, and its node feature matrix X, an oracle function f to label a query node, and a labelling budget B. We want to find a oracle function that:

$$\underset{f}{\operatorname{argmin}}\, Loss(GNN, f), \text{ when } |\mathcal{L}| = B \tag{4}$$

We propose a new metric, dissimilarity, in order to facilitate the performance of active learning protocols for graphs. The main difference between dissimilarity-based metrics and heterogeneity-based metrics is that heterogeneity-based metrics focus only on the unlabeled set of nodes and tries to choose a diverse set of nodes form the unlabeled set to be annotated, while dissimilarity-based metrics consider the labeled node set and unlabeled node set collectively. Given any particular labeled node set $\mathcal{L}_t$ at epoch t, we want our metric to select some different nodes from our specific $\mathcal{L}_t$. The concept of dissimilarity is crucial in many real world scenarios because in real world data sets, we often do not have control on the available labeled set of data before active labeling. Hence, design a metric that has strong correlation with what we currently have will be promising in selecting the most valuable next-sample to be labeled case by case.

We establish three dissimilarity score which directly evaluate the dissimilarity between the candidate nodes to be labeled and the nodes that have already been labeled.

3.1 Feature Dissimilarity Score

Feature dissimilarity score (FDS) is a metric to evaluate how dissimilar a candidate node is comparing to the nodes that have already been labeled in the node feature level. We adopt the idea of cosine similarity, a commonly used metric to evaluate how far away one vector is to another in multi-dimensional vector space. The cosine similarity of two non-zero vectors $\mathbf{X}$ and $\mathbf{Y}$ can be derived by using the following Euclidean dot product formula:

$$S_C(\mathbf{X}, \mathbf{Y}) = \cos(\theta) = \frac{\mathbf{X} \cdot \mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{5}$$

where X_i and Y_i are components of vector $\mathbf{X}$ and $\mathbf{Y}$ respectively.

To encourage selecting the nodes that are dissimilar from all the labeled nodes, the score $\phi_{\text{feature-dissimilarity}}$ then become:

$$\phi_{\text{feature-dissimilarity}}(v_i) = \frac{1}{\max_{l \in \mathcal{L}_t} S_C(v_i, l)} \tag{6}$$

where $\mathcal{L}_t$ is the labeled node set at epoch t.

3.2 Structural Dissimilarity Score

In the context of graph data sets, the similarity of two nodes is not only dependent on how similar their features are, but also dependent on their graph-level structures as well. To evaluate the structural dissimilarity accurately, we adopt the idea of second-order similarity proposed in LINE [9], that is, two nodes are similar if their neighbors are similar.

The matrix square of our adjacency matrix A provides us a simple closed-form solution for our structural dissimilarity score (SDS). The element $A^2_{i,j}$ gives the number of walks of length 2 from node i to node j, i.e. number of shared neighbors between node i and node j. Therefore, we could design our score $\phi_{\text{structural-dissimilarity}}$ to be:

$$\phi_{\text{structural-dissimilarity}}(v_i) = \frac{1}{\max_{l \in \mathcal{L}_t} A^2_{i,\text{index}(l)}} \tag{7}$$

where $\mathcal{L}_t$ is the labeled node set at epoch t and $\text{index}(l)$ is the index of the label node l.

Note that under the guidance of structural dissimilarity score, the more shared neighbours an unlabeled node has with those labeled nodes, the less likely that node is to be chosen for our next label. An intuitive interpretation of this score is that since we already have a labeled node to propagate the feature information to that large group of neighbors, the value of another label connected to those neighbors will be low in the setting of GNN models.

3.3 Embedding Dissimilarity Score

Embedding dissimilarity score (EDS) is designed to combine the feature information and structural information of the unlabeled candidate nodes. We use the embedding output from our GNN model and calculate the cosine similarity between those nodes and nodes with labels. Note that now the embedding consists both the feature information and structural information after the convolution of GNN.

$$\phi_{\text{embedding-dissimilarity}}(v_i) = \frac{1}{\max\limits_{l \in \mathcal{L}_t} S_C(\text{embedding}(v_i), \text{embedding}(l))} \tag{8}$$

where $\mathcal{L}_t$ is the labeled node set at epoch t and embedding(l) is the embedding of node l after the convolution.

3.4 Training

We follow the training process of the standard active learning protocol. At every training epoch t, if the labeling budget B is not reached, the AL Query Strategy module starts selecting the best candidate among the unlabeled nodes set $\mathcal{U}_t$, query it with the oracle, and add into the labelled nodes set $\mathcal{L}_t$ before epoch $t+1$ of the GNN training. The update policy of the lebelled set and unlabelled set strictly follows $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus v^*$ and $\mathcal{L}_{t+1} = \mathcal{L}_t \cup v^*$ where v^* is the node being selected. If the labeling budget B is reached, i.e., $|\mathcal{L}_t| = B$, then we stop selecting and the training is carried normally until GNN converges.

Our AL query module is described as follows:

$$\begin{aligned} v^* = \underset{i}{\text{argmax}}\ & \alpha\phi_{\text{entropy}}(v_i) + \beta\phi_{\text{density}}(v_i) \\ & + \gamma\phi_{\text{centrality}}(v_i) + \delta\phi_{\text{dissimilarity}}(v_i) \end{aligned}$$

Where $\phi_{\text{entropy}}(v_i)$, $\phi_{\text{density}}(v_i)$ and $\phi_{\text{centrality}}(v_i)$ are the three traditional AL query selection scores described in section 2.3 and $\phi_{\text{dissimilarity}}(v_i)$ is one of our three newly proposed dissimilarity scores. We will discuss more about the difference in effectiveness of these three scores in the experimental section.

The hyper-parameter α, β, γ and δ are time sensitive. We choose them in the following way:

$$\gamma \sim \mathbf{Beta}(1,\ 1.005 - Ct) \tag{9}$$

$$\alpha = \beta = \delta = \frac{1-\gamma}{3} \tag{10}$$

Here, t is the epoch number, **Beta** is the beta distribution where we sample γ from, C is a constant and a hyper-parameter to be chosen. On high level, our time-sensitive hyper-parameter set increase the weight of $\phi_{\text{centrality}}(v_i)$ at the early training stage because we want to collect more representitive nodes, and decrease the weight of $\phi_{\text{centrality}}(v_i)$ at the later training stage since we want to include more dissimilar nodes to our existing labelled set.

4 EXPERIMENTS AND DISCUSSION

4.1 Experimental Setting

We evaluate the performance of our proposed scores on the node classification tasks. We use three commonly used datasets: Cora, Citeseer, and Pubmed. For the Cora dataset, it consists of 2708 scientific publications that are classified into seven classes. And its graph has 5429 links. As for Citeseer, it has 3312 scientific publications classified into six classes. And its graph has 4732 links. On the other hand, we also have a larger dataset Pubmed which consists of 19717 publications that are classified into 3 categories. And it has 44338 links. We use GCN as our backbone module for active learning. More discussion about the backbone selection will be discussed in the next section. For all the baselines, we re-run their experiments to make sure we get reliable results.

We use Macro F1 and Micro F1, two classic classification evaluation methods measurements to assess the node classification performance. We use AGE as well as vanilla GCN where all labels are selected randomly, as our baselines. AGE is a strong baseline since it includes most of the existing active learning protocols by combining three heuristics including the entropy of the predicted label distribution, the node centrality score, and the distance between the node's embedding and its nearest clustering center (density score).

Under the active learning framework, we choose 4 labels for each class as our original labels as our seed set and start training our GCN module. Then, we incrementally add 1 additional label after each epoch based on our new proposed scores. We use the same total label budget B, which is 20 times the number of class to better compare with our baseline. After the budget is reached, we no long add more labels and continue training until converges.

Since the performance of active learning is often unstable, we run our experiments on 10 different validation sets using 10 different random seeds to reduce the variance of our results. We then take the average Macro F1 and Micro F1 over these 10 runs. For Pubmed, since the dataset is large, we only average over 3 runs due to time limit.

Here are some details of our hyper-parameter setting: We use the original implementation of GCN which has 2 graph convolution layers. We use a learning rate of 0.01 and a maximum number of epoches of 300. The hidden embedding dimension after layer 1 is 16. We choose our weight tuning hyper-parameter C to be 0.9 for Citeseer, 0.99 for Cora and 0.995 for Pubmed. The hyper-parameter C is chosen based on empirical results. One high level explanation of our choice of C is that Pubmed is denser than Cora, and Cora is denser than Citeseer. When the graphs are denser, we should put more weight on the centrality score, since the nodes with high centrality are more likely to be the most representation nodes in those graphs.

4.2 Results and Discussion

As we could observe from table 1, all three scores (FDS, SDS, and EDS) could improve the performance of node classification tasks on the three datasets. Comparing with the original AGE algorithm, the combination of FDS and SDS yields the best performance. The reason is that by combining FDS and SDS our algorithms utilized both the feature level information of the unlabeled nodes and the structural level information of the unlabeled nodes. Hence, we achieve heterogeneity in active learning by select the most dissimilar nodes comparing to our existing labels to be annotated. This will provide as much information as possible for our GCN backbone module when the budget is limited.

It is worth noticing that, although the EDS also combines both the structural and feature-level information of unlabeled nodes, it does not give as much performance gain as the other two scores. Moreover, in Pubmed, we observed a performance drop when using EDS as our additional score. This might be due to two reasons: first, just like the entropy score proposed by the original AGE model, the embedding dissimilarity score is also dependent on the quality of our GCN parameters. At the early stage of the training process, the GCN backbone might not be well trained, hence hard to provide an accurate embedding to aggregate the feature-level information and the structural information. Another reason is that the dimension of the output embedding is relatively small (equal to the number of classes in GCN), leading to dense vector distribution in high dimensions. Therefore, it might be hard to accurately evaluate the closeness between two embeddings by merely looking at their cosine similarity.

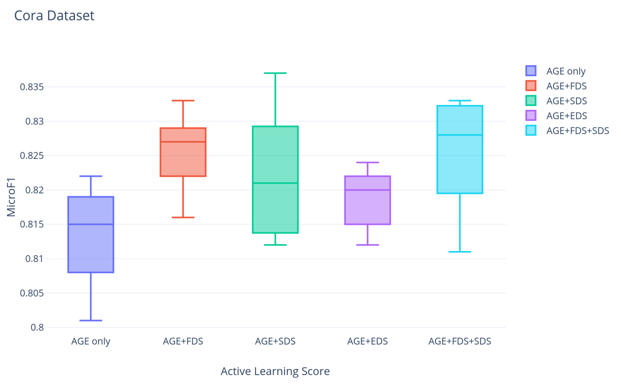

Figure 2: Cora Micro F1

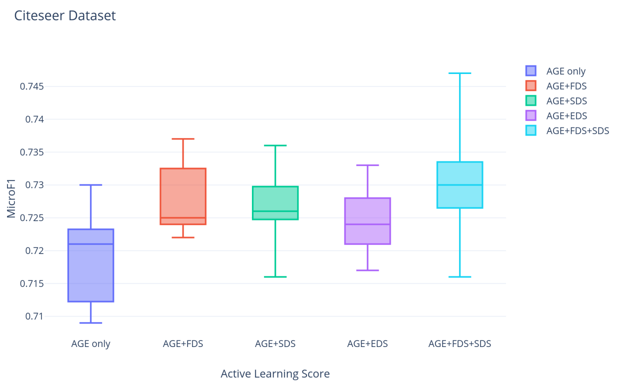

Figure 3: Citeseer Micro F1

Figure 2 and Figure 3 provide us with a closer look at how stable the performance of different algorithms are. As we could see, AGE with FDS performs consistently better than AGE only, and the variance is relatively small. Comparing with FDS, SDS are relatively more unstable. It could in some cases achieve superior performance than FDS, but perform worse in other cases (although better than the AGE baseline). It is likely those are the cases where the assumption in LINE [9] breaks, i.e., two nodes are very different in their features although they share the same neighbors. We suspect that cases will be more frequent in heterophilic graph setting [14], where two connected nodes are likely to be of different labels. We will discuss more about active learning for heterophilic graph datasets in the following sections.

4.3 Backbone Comparison

We also conduct experiments where we replace the vanilla GCN backbone [4] with other well-known graph neural networks. We use graph attention network (GAT) [10] and simplifying graph convolutional network (SGC) [12] as backbones while maintaining our original AL query metrics for GCN backbones. We run node classification tasks on both Cora and Citeseer data sets.

- **GAT** [10]: A novel neural network architecture that operates on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations.
- **SGC** [12]: A compact model which reduces excess complexity of GCN through successively removing nonlinearities and collapsing weight matrices between consecutive layers.

We employ the same scores, namely micro and macro F1 scores, to determine the performance of different active learning query metrics on GAT and SGC backbones. The results are shown in table 2 and table 3.

We observe that our three dissimilarity metrics could still improve the accuracy of node classification tasks when the base module is changed to GAT and SGC. This further assures the generalizability of our dissimilarity metrics among different GNN classifiers. Also, by comparing across different backbones, we find out that GCN yields better performance than GAT and SGC in almost all active learning settings. This further confirms our choice of GCN as the backbone module.

It is also notable that on Citeseer dataset, SGC resembles the performance of vanilla GCN while GAT have somewhat worse performance. We reason that in active learning setting, as the training set size is limited, attention based model like GAT are not able to properly learn the node representation. Thus, when it comes to entropy score and density score determined by embeddings of nodes, these two models naturally lost guidance, which interrupts the candidate node choice. This problem become significant on datasets like Citeseer where links are sparse and information aggregation is hard. On the other hand, SGC with its compactness does not require big training dataset which explains its similar performance to the vanilla shallow GCN. Thus, we can come to the conclusion that under the AGE structure, backbones that require a training set of decent size are not suitable.

4.4 A Closer Study on Heterophily

In recent years, the study of heterophilic graph mining has gain some research attention. The edge homophily ratio is the fraction of edges in a graph which connect nodes that have the same class label. Graphs with strong homophily have high edge homophily ratio (closer to 1), while graphs with strong heterophily have small edge homophily ratio (closer to 0). Zhu et. el. [14] have shown that algorithms perform well in homophilic graphs (e.g., Cora, Citeseer, Pubmed) might not perform as well in heterophilic datasets. Hence, it is worth checking the assumption of homophily in our active

Datasets	GCN	GCN+AGE	GCN+AGE+FDS	GCN+AGE+SDS	GCN+AGE+EDS	GCN+AGE+FDS+SDS
Cora \MacroF1 (%)	79.13	80.22	**81.54**	81.00	80.85	81.43
Citeseer \MacroF1 (%)	66.96	66.70	66.93	**67.86**	66.94	67.43
Pubmed \MacroF1 (%)	77.30	78.85	79.37	79.43	76.87	**79.53**
Cora \MicroF1 (%)	79.95	81.36	82.55	82.18	81.84	**82.59**
Citeseer \MicroF1 (%)	70.83	71.92	72.77	72.69	72.42	**73.06**
Pubmed \MicroF1 (%)	77.91	79.43	80.00	79.93	77.90	**80.23**

Table 1: F1 Scores of Different Active Learning Score Combination for GCN backbone

Datasets	GAT	GAT+AGE	GAT+AGE+FDS	GAT+AGE+SDS	GAT+AGE+EDS	GAT+AGE+FDS+SDS
Cora \MacroF1 (%)	77.61	78.95	80.08	80.45	79.37	**80.89**
Citeseer \MacroF1 (%)	62.05	61.96	61.90	63.22	61.95	**63.10**
Cora \MicroF1 (%)	78.28	80.39	80.93	81.76	80.46	**82.32**
Citeseer \MicroF1 (%)	65.76	67.65	68.12	68.50	67.92	**69.36**

Table 2: F1 Scores of Different Active Learning Score Combination for GAT backbone

Datasets	SGC	SGC+AGE	SGC+AGE+FDS	SGC+AGE+SDS	SGC+AGE+EDS	SGC+AGE+FDS+SDS
Cora \MacroF1 (%)	78.17	78.53	**80.58**	79.98	80.46	80.11
Citeseer \MacroF1 (%)	64.61	65.94	66.01	**67.25**	66.94	66.14
Cora \MicroF1 (%)	79.18	80.10	**81.87**	81.38	81.65	81.57
Citeseer \MicroF1 (%)	68.25	70.65	70.24	**71.72**	71.12	70.63

Table 3: F1 Scores of Different Active Learning Score Combination for SGC backbone

learning framework. In this way we could better study on the the generalizability of our protocol in the setting of heterophilic graphs.

- **Centrality Score**: The effectiveness of centrality score will somehow be affected by changing the assumption of homophily. While centrality usually implies representativeness on homophilic graphs, this might not be true on heterophilic graphs. In heterophilic graphs, all nodes with high centrality might belongs to a single class. For example, in a university enrolment network with professors and students, all high centrality nodes are professors.
- **Entropy Score**: The effectiveness of entropy score will not be affected by changing the assumption of homophily. The reason is that the entropy score is generated using the model output, and it aims for evaluating how uncertain the model is when regarding to a particular nodes. Changing the graph from homophilic to heterophilic will not affect the pipeline which chooses the most uncertain nodes.
- **Density Score and Embedding Dissimilarity Score**: The effectiveness of density score will not be affected if we change the assumption of homophily. This is because most density scores are generated using clustering algorithms in-taking model embedding outputs as raw inputs. If the model itself is well trained, the quality of the node embedding will be good enough to provide guidance for valuable node selection. The same arguments also applies for EDS.
- **Feature Dissimilarity Score**: The effectiveness of FDS will be severely affected by changing the assumption of homophily, since FDS is a score that completely neglects the structural information of a graph. While this is not a significant problem in homophilic graphs where nodes in different sub-graph clusters usually have different features, in heterophilic graphs, neglecting the structural information might lead to node selection within a single sub-graph cluster. This will severely affect the representativeness of nodes being selected.
- **Structural Dissimilarity Score**: The effectiveness of SDS will also be severely affected by changing the assumption of homophily, since the assumption "two nodes are similar if they share the same neighbors" will break under heterophilic setting.

To verify our hypothesises above, we conduct experiments on a heterophilic dataset, Chameleon, which contains 2277 nodes in 5 categories and 31421 edges. The data is collected by Rozemberczki et. al. from the English Wikipedia [7] which represent page-page networks on chameleons. Its homophily ratio is 0.23, which makes Chameleon a highly heterophilic dataset. For comparison, the homophily ratio is 0.81 in Cora, 0.74 in Citeseer and 0.80 in Pubmed. We use the same experimental settings as what we have for the other experiments. For fair comparison, we add only one score at a time when running on Chameleon dataset. We use GCN-Cheby

(GCC) to be our baseline, which is one of the SOTA models for node classification on heterophilic graphs [14].

Score	**GCC**	**GCC+Centrality**	**GCC+Entropy**
MacroF1 (%)	47.68	33.05	38.84
MicroF1 (%)	48.15	33.70	40.33

Score	**GCC+FDS**	**GCC+SDS**	**GCC+EDS**
MacroF1 (%)	25.10	32.28	39.99
MicroF1 (%)	30.99	32.58	40.41

Table 4: F1 Scores of Different Dissimilarity Scoring Functions on Chameleon dataset

The experimental results align with our hypothesis that FDS and SDS are the most severely affected scores when changed to heterophilic setting. Those two scores have the most performance drop compared to GCC. The performance drop of Centrality Score is also significant, but slightly better than FDS and SDS. In contrast, EDS and Entropy Score experience a relatively small performance drop compared to GCC, which means those two scores are more robust under heterophilic setting. It is also notable that none of those scores performs better that GCC, which means that there is still a significant room for performance improvement by finding specifically targeted types of active learning score functions for heterophilic graphs.

5 CONCLUSION AND FUTURE DIRECTIONS

5.1 Brief Summary

We propose a new metric, dissimilarity, in order to improve the effectiveness of active learning protocols for graphs. Our experimental results shows that all three types of scores will improve the performance of active learning protocols for graphs when added to existing active learning scores. We also show that a combination of FDS and SDS will yield the best performance since it aggregates both the node feature information and the graph structure information. Meanwhile, we provide a high level explanation about why those scores work. Then, we do some ablation study by replacing the GCN module with other similar GNN models, including GAT and SCC. We observe that GCN is the best candidate in our active learning setting for graphs. We also confirms the generalizability of our AL query metrics on other GNN classifiers. Lastly, we run experiments on heterophilic datasets and examine the effectiveness of different AL scoring functions on heterophilic datasets.

5.2 Future Directions

One possible future direction of our work is to find targeted types of active learning score functions for heterophilic graphs. We could also design an auto-tuning system that dynamically changes the distribution of the score functions with respective to different homophily scores.

Another possible future direction is to improve on the EDS. Our current EDS is inefficient since the model might not be well trained at the beginning. Future work could be carried on to overcome this bottleneck. Ideally, the performance of EDS itself should be as good as FDS + CDS if our embedding is good, since it utilizes both the feature-level and the structural information.

A third possible future direction is to consider developing an efficient performance-based active learning algorithm. Currently we are not using any performance-based active learning algorithm since existing performance-based algorithms often require a lot of training time (e.g. computing the expected model change using reinforcement learning). We could try to find an efficient way to evaluated the expected performance gain when adding a particular label and integrate this into our unified active learning framework.

6 TASK DISTRIBUTION

Our task distribution could be seen on table 6. Also, we would like to specially thank Yewen Wang and Prof. Yizhou Sun for their guidance and generous help on the course of this project.

REFERENCES

[1] Hongyun Cai, Vincent Wenchen Zheng, and Kevin Chen-Chuan Chang. 2017. Active Learning for Graph Embedding. *CoRR* abs/1705.05085 (2017). arXiv:1705.05085 http://arxiv.org/abs/1705.05085

[2] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph Policy Network for Transferable Active Learning on Graphs. *CoRR* abs/2006.13463 (2020). arXiv:2006.13463 https://arxiv.org/abs/2006.13463

[3] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph Policy Network for Transferable Active Learning on Graphs. https://doi.org/10.48550/ARXIV.2006.13463

[4] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[5] Jiaqi Ma, Ziqiao Ma, Joyce Chai, and Qiaozhu Mei. 2022. Partition-Based Active Learning for Graph Neural Networks. *arXiv preprint arXiv:2201.09391* (2022).

[6] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/ Previous number = SIDL-WP-1999-0120.

[7] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. arXiv:1909.13021 [cs.LG]

[8] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[9] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ accepted as poster.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. https://arxiv.org/abs/1710.10903

[12] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 6861–6871.

[13] Yuexin Wu, Yichong Xu, Aarti Singh, Yiming Yang, and Artur Dubrawski. 2019. Active learning for graph neural networks via node feature propagation. *arXiv preprint arXiv:1910.07567* (2019).

[14] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.

Task	**Student**
Discussion and problem formulation	Zhicheng, Xiaxuan, Yuxin and Yifu
Implementation of AL scoring functions	Zhicheng
Implementation of base modules	Yifu
Processing and experiments on heterophilic datasets	Yuxin
Experiments for KMedoids clustering algorithms	Xiaxuan
Written reports	Zhicheng, Xiaxuan, Yuxin and Yifu

Table 5: Task Distribution