2022 IEEE Eighth International Conference on Big Data Computing Service and Applications (BigDataService) | 978-1-6654-5890-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigDataService55688.2022.00022

Active Learning for Node Classification using a Convex Optimization approach

Deepesh Agarwal
Department of Electrical and Computer Engineering
Kansas State University
Manhattan, Kansas, USA
deepesh@ksu.edu

Balasubramaniam Natarajan
Department of Electrical and Computer Engineering
Kansas State University
Manhattan, Kansas, USA
bala@ksu.edu

***Abstract*—The recent advancements related to big data analytics in the era of Industry 4.0 are fueled by development and deployment of decision models based on neural network (NN) architectures. In addition to the data represented in Euclidean space, like, images, text, or videos, there are increasing applications which demand data representation in non-Euclidean domains. Such data are typically represented as graphs with complex interactions and interdependencies between various entities. The complexity of graph data imposes substantial challenge on the traditional Deep NN models, and Graph Neural Networks (GNN) are a powerful extension for modeling and analysis of networked data. The training of these computational models requires large amounts of labeled data. Active Learning (AL) helps to overcome this issue by selecting the most informative instances for labeling during the training process. This paper combines AL with GNN for semi-supervised classification of nodes in attributed graphs. The AL framework is portrayed as a convex optimization problem by employing Dissimilarity-based Sparse Modeling Representative Selection (DSMRS). The experimental evaluation demonstrates that using the selected graph-specific metrics (centrality and robustness measures) as AL heuristics leads to an improvement in classification performance by upto 10%.**

***Index Terms*—Active Learning, Graph Neural Networks, Convex Optimization, Node Classification, Graph-specific metrics**

I. Introduction

In recent years, there has been a substantial development in Artificial Intelligence (AI) engines and machine learning technologies to harness the power of data and build efficient decision models that are capable of making accurate predictions and inferences. However, a huge amount of labeled data is required to train these computational models. Semi-supervised learning approaches like Active Learning (AL) helps to address this limitation by actively and strategically choosing the most informative samples from the pool of unlabeled data. Consequently, it is possible to accomplish a better predictive performance with a significantly reduced training sample size.

There is an increasing research interest in analyzing and learning from data in non-Euclidean space, like graphs. It has a wide variety of applications in several interesting areas like social networks, knowledge graphs, protein-protein interaction networks, combinatorial optimization and molecular fingerprinting. Graph Neural Networks (GNN) provide an elegant framework for effective inferencing on such graph-structured data. Given the extensive applications of learning from graph-structured data and the benefits of semi-supervised learning techniques, it would be highly beneficial to consolidate AL with GNN for many downstream tasks like node classification, link prediction, community detection and graph classification. This article presents an AL framework combined with GNN for node classification in attributed graphs using a convex optimization approach.

A. Related Work

There is only a limited number of AL models for graph-structured data as compared to other types of data that are represented in Euclidean space, like images, text or numerical data [1]. It has also been established that the use of graph theoretic measures as acquisition functions in AL provide better performance than using traditional heuristics like classifier uncertainty, entropy of class probabilities or classification margin [2]. The early works related to application of AL on graph data are based on cluster assumption and local and global consistency [3], [4]. In these techniques, it is assumed that nearby points on the same structure (i.e., cluster or manifold) are likely to have the same labels. This category of AL methodologies do not mimic realistic settings and restricts the modeling capacity.

Some of the works related to implementing AL of graph-structured data are linked to earlier classification models like Gaussian random fields [5], [6], [7]. However, these models do not incorporate node features and label information in the learning process. Moreover, some of these models are not adaptive in the sense that the active learner is not updated based on the newly labeled instances. These limitations are addressed by approaches that employ AL models coupled with GNN architectures [8], [2]. However, such studies are very limited and the corresponding acquisition functions are based only on matrix concepts for centrality formulation. This results in a lack of intuition related to the underlying network topology.

In this article, we present a convex optimization driven AL framework combined with GNN that uses four graph theoretic measures, namely, eigenvector centrality, closeness centrality, betweenness centrality and Effective Graph Resistance (EGR) as AL heuristics for active node selection in attributed graphs.

DOI 10.1109/BigDataService55688.2022.00022

B. Contributions

This work addresses the node classification task in attributed graphs by utilizing a framework that combines AL and GNN using a rigorous convex optimization approach. The novel contributions of this paper are as follows:

- Four graph-theoretic metrics are used as AL heuristics/acquisition functions for active selection of nodes.
- Both node attributes and topological information are incorporated in the learning scheme. The node features are exploited while training the GNN-based decision model and topological information is considered during selective sampling of the nodes.
- An inductive GNN approach is used for building node classification models. This allows the approach to be scalable across graphs of different sizes as well as subgraphs within the given graph.

The remainder of this article is organized as follows: background on GNN and AL via convex optimization is provided in Section II, and convexity analysis is presented in Section III. Section IV elaborates upon the proposed methodology, followed by experimental evaluation in Section V. The article ends with concluding remarks in Section VI.

II. Preliminaries

A. Graph Neural Networks

Recently, there has been a surge of interest in learning with graph-structured data, like, biological, financial and social networks, and knowledge graphs. There are several advantages of representing data in form of graphs. Firstly, they provide an intuitive approach to visualize concepts of interactions and relationships. Additionally, it allows us to elucidate a complex problem by transforming it into simpler representations. The concepts of graph theory can be exploited to model and analyze such forms of data representations. However, it is difficult to analyze and interpret graph-structured data using traditional Deep Neural Network (DNN)-based learning algorithms. This is primarily because graphs cannot be described in Euclidean space, which makes it difficult to interpret as compared to other types of data like images or text. Furthermore, the irregular structure of graphs, inconsistent size of unordered nodes and variable neighborhood configuration of the nodes prevents crucial mathematical operations like convolutions to be applied on graph-structured data. GNN help to overcome these limitations by extending DNN architectures for graph domain.

GNNs are typically based on recursive message passing (i.e., neighborhood aggregation), wherein every node in the graph aggregates attributes of its neighbors in order to compute its own feature vector [9], [10]. This allows a node to be represented by its transformed feature vector or node embedding after a specific number of aggregation iterations, thereby capturing the structural information across its p-hop neighborhood. Several variants of GNN with diverse neighborhood aggregation functions and different pooling schemes have been presented in the literature [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [9], [21], [22], [23], [24], [25]. These GNN methodologies have been empirically shown to achieve state-of-the-art performance on many downstream tasks such as link prediction, node classification and graph classification. In this work, we use GraphSAGE [15], an inductive node embedding approach that concurrently learns both the topological structure and distribution of features for a node in its local neighborhood. The operation executed at i^{th} node embedding layer is given by eq. (1).

$$\begin{aligned} h_u^{(i)} &= f^{(i)}\left(h_u^{(i-1)}, h_{N(u)}^{(i-1)}\right) \\ &= g\left[\theta_C^{(i)} h_u^{(i-1)} + \theta_A^{(i)} \tilde{A}\left(h_{N(u)}^{(i-1)}\right)\right] \end{aligned} \tag{1}$$

Here, $h_u^{(i)}$ represents the node embedding of node u at i^{th} layer; $\tilde{A}$ denotes the aggregation operation; θ_C and θ_A are the parameters of the combination and aggregation operation of GNN, respectively; $N(u)$ describes the neighborhood of node u and $g[\cdot]$ denotes the activation function.

B. Active Learning via Convex Optimization

AL constitutes a category of machine learning techniques where the fundamental premise is to improve the performance of a leaner by iteratively querying (i.e., seeking labels for selected unlabeled samples in the dataset) the Oracle (i.e., external source of labeling, like the domain experts) [26]. The key idea is to allow the learner to be active and collaborate with the Oracle to obtain labels for most informative samples, thereby achieving better performance with much fewer labelled training instances as compared to conventional supervised machine learning techniques. Such a semi-supervised learning strategy is very beneficial in applications exploiting contemporary machine learning pipelines, where the labeling procedure might be expensive, difficult or time-consuming [27]. In this work, we employ AL strategy along with GNN for node classification in attributed graphs. We use an AL framework based on convex programming [28] that leverages Dissimilarity-based Sparse Modeling Representative Selection (DSMRS) algorithm [29], [30] for selecting most informative nodes in the given graph-structured data. There are numerous advantages of using a convex optimization-based AL framework as compared to the traditional iterative querying mechanism [28]. Firstly, it incorporates aspects of both uncertainty sampling as well as sample diversity while formulating the convex optimization problem. This allows to select multiple samples from the dataset which are not only informative but are also diverse with respect to each other. Moreover, the data distribution is assimilated via a dissimilarity matrix in the problem formulation. The formulation of AL as a convex optimization problem is discussed next.

Given a graph G with N nodes and dissimilarity matrix $\boldsymbol{D} = \{d_{ij}\}_{i,j=1,2,3,\ldots,N}$, the goal is to find a small subset of nodes that are well representative of the entire graph. Here, d_{ij} represents how well the node j is represented by another node i. The smaller values of d_{ij} denotes better representation

and vice-versa. The dissimilarity values are assumed to be non-negative and $d_{jj} \leq d_{ij}$ for all i, j. In order to find the representative nodes of the graph, $\boldsymbol{Z} = \{z_{ij}\}_{i,j=1,2,3,...,N}$ is introduced as an optimization variable. $z_{ij} \in [0,1]$ is associated to d_{ij} and indicates the probability of node i being a representative of node j. The optimization function consists of two components: (i) encoding cost of all N nodes using the representative nodes, and (ii) penalizing the number of selected representatives. The encoding cost of node j via i can be expressed as $d_{ij}z_{ij} \in [0, d_{ij}]$. So, the total encoding cost for all the nodes in the graph is given by eq. (2).

$$\sum_{i,j} d_{ij} z_{ij} = \text{tr}(\boldsymbol{D}^T \boldsymbol{Z}) \tag{2}$$

The number of representative nodes can be directly associated to the the number of non-zero rows in $\boldsymbol{Z}$ [29]. Consequently, a convex surrogate for the cost related to number of selected representatives is given by eq. (3), where $q \in \{2, \infty\}$.

$$\sum_{i=1}^{N} ||z_i||_q \triangleq ||\boldsymbol{Z}||_{q,1} \tag{3}$$

Combining both the components in eqs. (2) and (3) together, the overall convex optimization problem is given in eq. (4).

$$\min \lambda||\boldsymbol{Z}||_{q,1} + \text{tr}(\boldsymbol{D}^T \boldsymbol{Z}) \text{ s.t. } \boldsymbol{Z} \geq 0, \mathbf{1}^T \boldsymbol{Z} = \mathbf{1}^T \tag{4}$$

Here, the λ parameter balances the costs associated with encoding and number of representatives, and controls the batch size in AL. A smaller value of λ lays more importance on better encoding, thereby obtaining more representative nodes. On the other hand, a higher value of λ corresponds to more emphasis on penalizing, thereby obtaining less representative nodes. The constraints ensure that each column of $\boldsymbol{Z}$ is a probability vector.

This DSMRS-based problem formulation can be well extended to AL by incorporating sample informativeness score and sample diversity score. The informativeness score for a node u, $s_{inf}(u) \in [0,1]$, represents the degree of its importance or informativeness. A higher value of $s_{inf}(u)$ signifies that that node u is highly representative of the given graph. On the other hand, a lower value of $s_{inf}(u)$ represents that node u is not so representative and conveys less information about the given graph. The diversity score for a node u is given by eq. (5).

$$s_{div}(u) = \frac{\min_{j \in \mathcal{L}} d_{ju}}{\max_{k \in \mathcal{U}} \min_{j \in \mathcal{L}} d_{jk}} \tag{5}$$

Here, $\mathcal{L}$ and $\mathcal{U}$ are the sets of indices of labeled and unlabeled nodes respectively.

If the closest labeled node to an unlabeled node u is very similar to it, $s_{div}(u) \to 0$, and selecting such a node doesn't promotes diversity. On the contrary, when all the labeled nodes are very dsisimilar from an unlabeled node u, $s_{div}(u) \to 1$, and selecting such a node for labeling would be beneficial as it would be different from the other labeled nodes. The combined score matrix ($\boldsymbol{S}$) is given by eq. (6), where the overall score for each node $s(u_i)$ is evaluated as shown in eq. (7).

$$\boldsymbol{S} = \text{diag}\left(s(u_1), s(u_2), s(u_3), ..., s(u_{|\mathcal{U}|})\right) \tag{6}$$

$$s(u_i) = \max\{s_{inf}(u_i), s_{div}(u_i)\} \tag{7}$$

Other mathematical operations like mean or minimum can be used instead of max to compute the overall score for each node in eq. (7). Finally, AL process can be represented in the form of an optimization problem as shown in eq. (8).

$$\min \lambda||\boldsymbol{SZ}||_{q,1} + \text{tr}(\boldsymbol{D}^T \boldsymbol{Z}) \text{ s.t. } \boldsymbol{Z} \geq 0, \mathbf{1}^T \boldsymbol{Z} = \mathbf{1}^T \tag{8}$$

If an unlabeled node u_i has lower score $s(u_i)$, the optimization program puts less penalty on the i^{th} row of $\boldsymbol{Z}$ being non-zero, and vice-versa. In the next section, we will analyze the convexity of the optimization problem (8).

III. Convexity Analysis

The following result formalises the convexity of the optimization problem for Active Learning.

Theorem 1. *The optimization problem for Active Learning, given by:*

$$\textit{min } \lambda||\boldsymbol{SZ}||_{q,1} + \textit{tr}(\boldsymbol{D}^T \boldsymbol{Z}) \textit{ s.t. } \boldsymbol{Z} \geq 0, \mathbf{1}^T \boldsymbol{Z} = \mathbf{1}^T$$

over the optimization variable $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$ is convex, where $\boldsymbol{D}$ is the dissimilarity matrix indicating the degree of dissimilarity between samples (nodes) in the dataset (graph), $\boldsymbol{S}$ is the overall score matrix formed by combining informativeness and diversity scores of all the samples in the dataset and λ is a parameter that balances the costs associated with encoding all the samples and number of representatives.

Proof. It is well known that a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if dom(f) is a convex set and if for all x_1, $x_2 \in$ dom(f), and $0 \leq \theta \leq 1$, we have

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2) \tag{9}$$

Consider $||\boldsymbol{SZ}||$

$$||\boldsymbol{S}(\theta \boldsymbol{A}_1 + (1-\theta)\boldsymbol{A}_2)|| = ||\theta \boldsymbol{SA}_1 + (1-\theta)\boldsymbol{SA}_2|| \tag{10}$$

By triangle inequality for matrix norms,

$$\begin{aligned} ||\boldsymbol{S}(\theta \boldsymbol{A}_1 + (1-\theta)\boldsymbol{A}_2)|| &\leq ||\theta \boldsymbol{SA}_1|| + ||(1-\theta)\boldsymbol{SA}_2|| \\ &= \theta||\boldsymbol{SA}_1|| + (1-\theta)||\boldsymbol{SA}_2|| \end{aligned} \tag{11}$$

This indicates that $||\boldsymbol{SZ}||$ is convex. Further, λ is a positive constant multiplier. Hence, $\lambda||\boldsymbol{SZ}||$ is also convex.

Consider the second term, $\text{tr}(\boldsymbol{D}^T \boldsymbol{Z}) = \sum_{i,j} d_{ij} z_{ij}$.

$$\begin{aligned}\text{tr}(\boldsymbol{D}^T(\theta\boldsymbol{A}_1+(1-\theta)\boldsymbol{A}_2)) &= \sum_{i,j} d_{ji}(\theta a_1+(1-\theta)a_2)_{ji}\\ &\leq \sum_{i,j}\theta d_{ji}a_{1ji}+\sum_{i,j}(1-\theta)d_{ji}a_{2ji}\\ &= \theta\sum_{i,j} d_{ji}a_{1ji}+(1-\theta)\sum_{i,j} d_{ji}a_{2ji}\\ &= \theta\,\text{tr}(\boldsymbol{D}^T\boldsymbol{A}_1)+(1-\theta)\,\text{tr}(\boldsymbol{D}^T\boldsymbol{A}_2)\end{aligned} \tag{12}$$

This indicates that $\text{tr}(\boldsymbol{D}^T\boldsymbol{Z})$ is convex. Since both the terms of eq. (8) are convex, their sum is also convex. The linear equality constraints ensures that the overall formulation in eq. (8) is convex. □

IV. Methodology

This work proposes an AL framework for active node classification on attributed graphs. The proposed framework is shown in Figure 1. The raw input graph-structured data is pre-processed so as to convert it into a format suitable for further operations and analysis. After appropriate preprocessing steps, training, validation and test masks are generated to split the original graph-structured dataset into respective components. This is followed by definition of GNN model architecture (no. of GNN layers, no. of feed-forward layers, hidden dimensions, dropout, optimizer, epochs, etc.) and training an initial model using the initial labeled dataset. The dissimilarity matrix $\boldsymbol{D}=\{d_{ij}\}_{i,j=1,2,3,\ldots,N}$ is evaluated by computing normalized Euclidean distances between node embeddings of all node pairs i, j.

The selection of representative nodes is initiated by calculation of informativeness ($s_{inf}(u)$) and diversity ($s_{div}(u)$) scores for all the nodes in the unlabeled pool. $s_{inf}(u)$ is obtained based on the values of AL heuristics. This step is similar to computing heuristics like classifier uncertainty, classification margin or entropy of class probabilities in conventional iterative querying AL frameworks [31], [32]. In this work, we use the following graph theoretic measures as acquisition functions to select the representative nodes in the graph-structured data.

1) Eigenvector centrality
   The most informative node from the unlabeled pool of nodes is given as:
$$u^* = \arg\max_u \frac{1}{\kappa}\sum_{j=1}^{N} A_{uj}c_j \tag{13}$$
   Here, c_j is the centrality of node j, A_{uj} ensures that only the neighbors of node u contribute to the sum and κ is the largest eigenvalue of A.
   This metric tries to select the node which is connected to many other nodes in the network, while simultaneously weighing all the neighbors based on their importance or influence. It overcomes the limitation of degree centrality measure (already used in the literature [1]), which considers equal weights for all the neighbors of the node.
2) Closeness centrality
   The most informative node is selected using:
$$u^* = \arg\max_u \frac{N}{\sum_{j=1}^{N} d_{uj}} \tag{14}$$
   Here, d_{uj} is the shortest distance between the nodes u and j.
   The closeness centrality metric tries to select the node whose average distance to other nodes in the graph is smaller.
3) Betweenness centrality
   The most representative node is expressed as:
$$u^* = \arg\max_u \sum_{s=1}^{N}\sum_{t=1}^{N} n_{st}^u \tag{15}$$
$$n_{st}^u = \begin{cases} 1, \text{if } u \text{ lies on the shortest path from } s \text{ to } t \\ 0, \text{otherwise}\end{cases} \tag{16}$$
   This measure selects the nodes which have maximum control over information passing between other nodes in the graph.
4) Effective Graph Resistance (EGR)
   The most informative node from the unlabeled pool of nodes is selected using:
$$u^* = \arg\max_u (R_G - R_{Gu}) \tag{17}$$
   Here, R_G is the EGR of complete graph G and R_{Gu} represents the EGR of G after removal of node u.
   This metric leads to selection of the nodes whose removal maximizes the decrease in graph robustness.

The aforementioned graph theoretic measures have not been used as AL heuristics/acquisition functions in the literature of node classification using active learning, and is a novel contribution of this work. The advantages of the proposed AL framework are multifold. Firstly, it incorporates both node attributes and topological information in the learning scheme. The node features are used while training the GNN-based decision model and topological information is considered during selective sampling of the nodes. Unlike a majority of existing works in the literature, we use inductive GraphSAGE [15] approach for building node classification models. This allows the approach to be scalable across graphs of different sizes as well as subgraphs within the given graph.

V. Experimental Evaluation

A. Datasets Description

The proposed AL framework is evaluated by performing experiments over two datasets, namely, Cora [33] and CiteSeer [34]. These are the citation networks consisting of scientific publications. Every node in the graphs represents a publication characterized by a set of binary features indicating the presence/absence of unique words from the dictionary. The edges in the graphs signify citation links among different publications. The properties of the datasets are summarized in Table I.

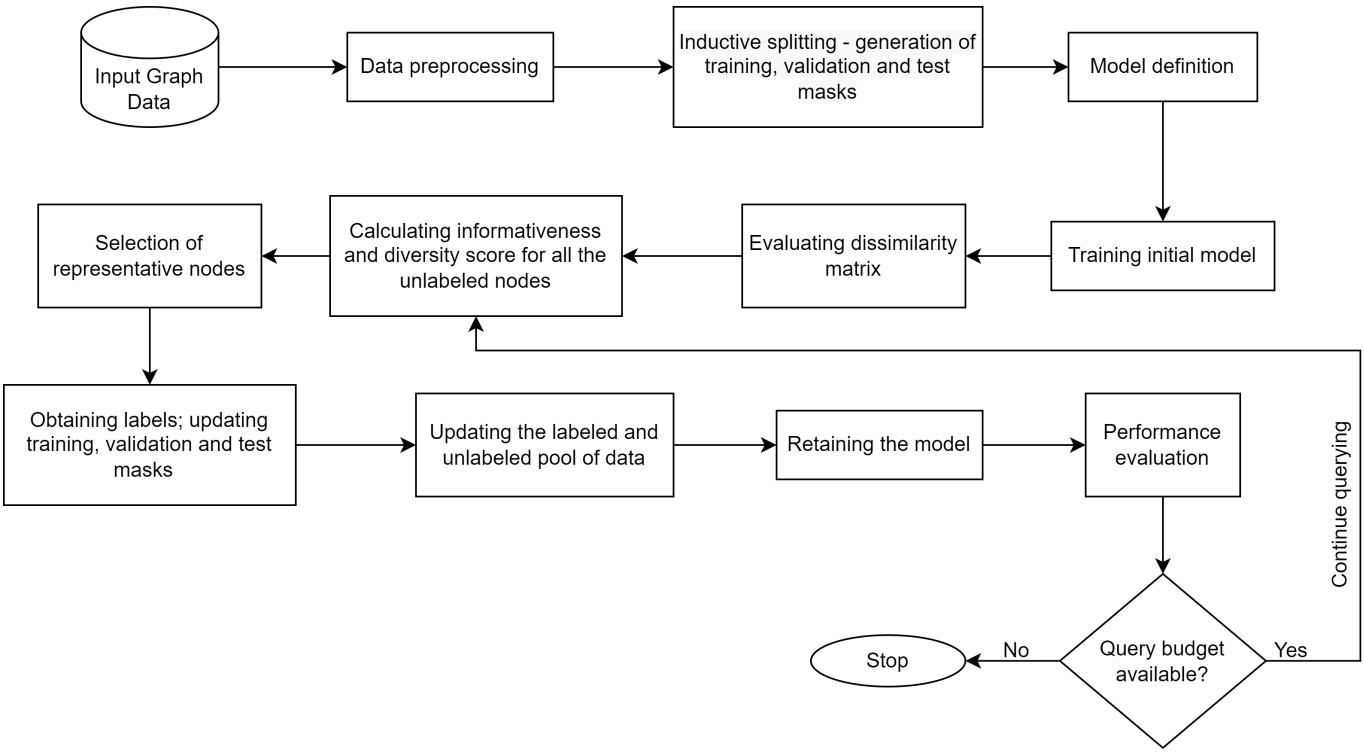

Fig. 1. Proposed Active Learning framework.

TABLE I
SUMMARY OF THE DATASETS CONSIDERED. AVG. DEG.: AVERAGE DEGREE, AVG. CC: AVERAGE CLUSTERING COEFFICIENT

Dataset	Nodes	Links	Features	Classes	Avg. Deg.	Avg. CC
Cora	2708	5429	1433	7	4.00	0.24
CiteSeer	3312	4732	3703	6	2.84	0.17

B. Experimental Settings

All the downstream tasks related to representing and processing graphical data are handled using NetworkX [35] library in Python. PyTorch [36] is used to implement aspects of deep learning and GNN-specific tasks are integrated using Deep Graph Library (DGL) [37]. The detailed architecture of the GNN is as follows: depth (i.e., number of node embedding modules): 2; number of neurons in 2 layers: 48, 48; number of multi-layer perceptron (MLP) layers: 2; activation function: ReLU (except last layer with softmax); aggregation function: mean.

The training of GNNs is carried out in a mini-batch manner. 1.5 - 2% of the total nodes in the graph-structured datasets are randomly chosen to construct the initial labeled dataset. Around 20 - 25% of the nodes (uniformly selected over all classes) are kept aside for testing and are not used during training and validation steps. We make a total of 100 queries on Cora and 120 queries ($\sim$ 3.5 - 4% of the total nodes) on CiteSeer datasets. AL models are trained by considering four different graph-theoretic measures, namely, eigenvector centrality, closeness centrality, betweenness centrality and EGR (described in Section IV) as heuristics for selecting the most informative nodes. The evaluation is repeated for 100 times and the average values of classification accuracy scores are reported in Section V-C. The degree centrality and clustering coefficient (CC) metrics, used in [1] are treated as the baselines. Degree centrality of a node u represents the fraction of nodes in the networks that are connected to u. The use of this metric as an AL heuristic results in selection of a node that is connected to maximum number of nodes in the network. On the contrary, CC of a node u is evaluated by computing the fraction of triangles possible through it. When used an acquisition fuction in AL framework, it tries to select nodes that possess high tendency to form clusters together.

C. Results and Discussion

The values of classification accuracy using all the four graph-theoretic measures (eigenvector centrality, closeness centrality, betweenness centrality and EGR) as well as the baselines (degree centrality and clustering coefficient) for Cora and CiteSeer datasets are tabulated in Table II and Table III respectively. The corresponding plots are shown in Figure 2 and Figure 3 respectively. It can be observed that two of the metrics, i.e., betweenness centrality and EGR consistently exhibit better performance as compared to both the baselines for Cora and CiteSeer datasets.

Degree centrality is a naive graph theoretic measure which quantifies the number of connections of a node to other nodes in the graph. On the other hand, the metrics like betweenness centrality and EGR incorporate topological information in a

TABLE II
RESULTS FOR CORA DATASET

Query Strategy	Initial Classification Accuracy	Classification Accuracy after no. of queries			
		25	50	75	100
Degree	29.43%	46.42%	58.57%	68.28%	75.14%
Clustering Coefficient	27.66%	43.51%	52.17%	65.96%	69.86%
Eigenvector	32.64%	45.79%	64.43%	75.84%	77.92%
Closeness	29.71%	42.06%	50.28%	54.42%	60.97%
Betweenness	*30.42%*	*54.71%*	*67.85%*	*70%*	*80.57%*
EGR	*31.75%*	*52.36%*	*68.79%*	*75.53%*	*83.81%*

TABLE III
RESULTS FOR CITESEER DATASET

Query Strategy	Initial Classification Accuracy	Classification Accuracy after no. of queries			
		25	50	75	100
Degree	27.64%	34.06%	46.18%	60.69%	71.56%
Clustering Coefficient	24.33%	37.57%	40.82%	58.48%	63.12%
Eigenvector	28.14%	39.51%	50.42%	67.31%	75.73%
Closeness	26.53%	35.56%	44.10%	62.54%	72.02%
Betweenness	*30.27%*	*41.78%*	*52.37%*	*63.22%*	*77.82%*
EGR	*30.91%*	*43.88%*	*56.25%*	*68.48%*	*79.38%*

more elegant way as compared to degree centrality. In the context of AL framework, betweenness centrality tries to select the node having maximum control over information passing between other nodes in the graph. EGR is a robustness measure which picks up the nodes whose removal maximizes the decrease in graph robustness. Therefore, the use of these graph theoretic metrics as AL heuristics leads to an improvement in classification performance by upto 10% as compared to the baseline.

VI. CONCLUSION

This article presents an AL framework to address node classification task in attributed graphs using a convex optimization approach. The proposed method integrates both topological information as well as node attributes within the learning scheme. The decision models are trained and updated using GraphSAGE, an inductive node embedding approach that learns both the topological structure and distribution of features for a node in its local neighborhood. Graph theoretic measures are used as AL heuristics to select the most informative nodes in the graph for annotation. It is observed that betweenness centrality and EGR consistently outperform the baseline and improve the classification performance by upto 10%. Future extensions of this work include exploring the utility of other graph theoretic measures and evaluating the proposed framework over larger graphs as well as graphs from different application domains.

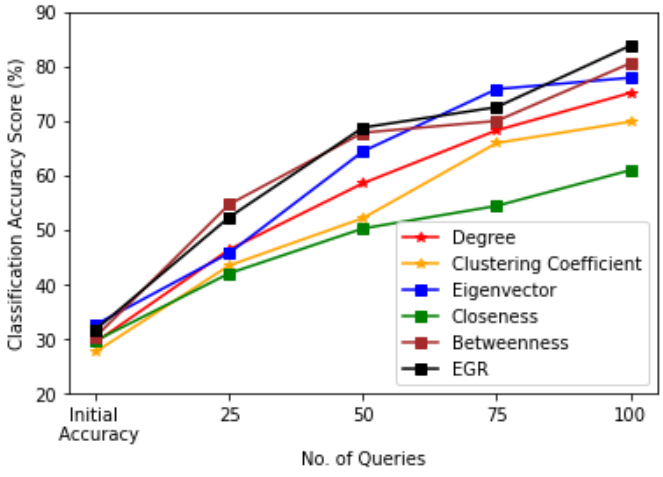

Fig. 2. Plots of classification accuracy scores vs. no. of queries for Cora dataset.

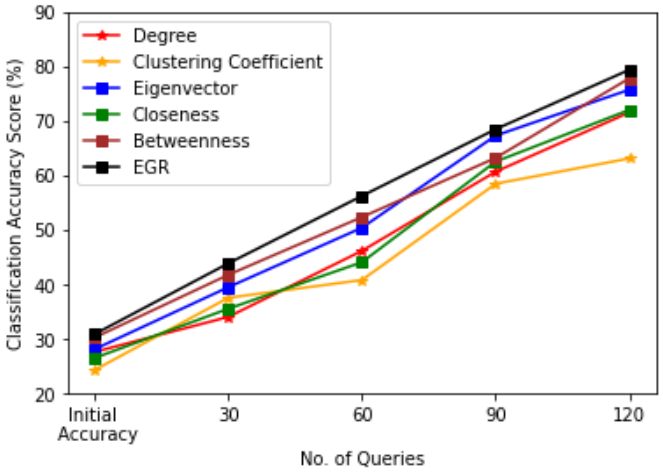

Fig. 3. Plots of classification accuracy scores vs. no. of queries for CiteSeer dataset.

REFERENCES

[1] K. Madhawa and T. Murata, "Active learning for node classification: An evaluation," *Entropy*, vol. 22, no. 10, p. 1164, 2020.

[2] L. Gao, H. Yang, C. Zhou, J. Wu, S. Pan, and Y. Hu, "Active discriminative network representation learning," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[3] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[4] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, 2003.

[5] Q. Gu and J. Han, "Towards active learning on graphs: An error bound minimization approach," in *2012 IEEE 12th International Conference on Data Mining*, IEEE, 2012, pp. 882–887.

[6] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *ICML*, 2010.

[7] M. Ji and J. Han, "A variance minimization criterion to active learning on graphs," in *Artificial Intelligence and Statistics*, PMLR, 2012, pp. 556–564.

[8] H. Cai, V. W. Zheng, and K. C.-C. Chang, "Active learning for graph embedding," *arXiv preprint arXiv:1705.05085*, 2017.

[9] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5453–5462.

[10] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, PMLR, 2017, pp. 1263–1272.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[12] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, *et al.*, "Interaction networks for learning about objects, relations and physics," *Advances in neural information processing systems*, vol. 29, 2016.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[14] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[16] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, no. 8, pp. 595–608, 2016.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *International Conference on Learning Representations (ICLR)*, 2016.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations (ICLR)*, 2018.

[20] A. Santoro, D. Raposo, D. G. Barrett, *et al.*, "A simple neural network module for relational reasoning," *Advances in neural information processing systems*, vol. 30, 2017.

[21] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap, "Measuring abstract reasoning in neural networks," in *International conference on machine learning*, PMLR, 2018, pp. 511–520.

[22] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[23] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[24] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 7134–7143.

[25] J. You, J. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," *arXiv preprint arXiv:2101.10320*, 2021.

[26] B. Settles, "Active learning, volume 6 of synthesis lectures on artificial intelligence and machine learning," *Morgan & Claypool*, 2012.

[27] ——, "Active learning literature survey," 2009.

[28] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sasrty, "A convex optimization framework for active learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 209–216.

[29] E. Elhamifar, G. Sapiro, and R. Vidal, "Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[30] E. Elhamifar, G. Sapiro, and S. S. Sastry, "Dissimilarity-based sparse subset selection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2182–2197, 2015.

[31] D. Agarwal, P. Srivastava, S. Martin-del-Campo, B. Natarajan, and B. Srinivasan, "Addressing uncertainties within active learning for industrial iot," in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, IEEE, 2021, pp. 557–562.

[32] ——, "Addressing practical challenges in active learning via a hybrid query strategy," *arXiv preprint arXiv:2110.03785*, 2021.

[33] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[34] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.

[35] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[36] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[37] M. Wang, D. Zheng, Z. Ye, *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.