Active Learning for Attributed Graphs

Florence Robert-Regol

Department of Electrical & Computer Engineering, McGill University, Montréal

July 2020

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

ACKNOWLEDGEMENTS

I would like to thank everyone who have supported me for the duration of my Master's degree. In particular, I would like to express my gratitude toward my supervisor Professor Mark Coates. As his student, I was given multiple opportunities to learn and develop valuable research skills. I am also grateful for the insightful feedback and guidance he provided me throughout my studies. His supervision contributed greatly in making my experience as a graduate student enjoyable and rewarding, and motivated me to pursue in academia. I thank my colleague Soumyasundar Pal (PhD candidate), for his contribution to this work and his useful input, and everyone from the computer networks research lab of McGill for fostering a supportive and stimulating environment. I also appreciate the support and guidance provided by the department of Electrical and Computer Engineering of McGill University. Lastly I would like to thank my family for all their help and encouragements.

ABSTRACT

Node classification in attributed graphs is an important task in multiple practical settings, but it can often be difficult or expensive to obtain labels. Active learning is an approach that aims to improve performance for a limited budget of labels by targeting informative nodes that will form the labelled set. As such, adopting active learning for node classification in attributed graphs seems to be a promising avenue, and is the topic of this thesis. The best existing methods are based on graph neural networks, but they often perform poorly unless a sizeable validation set of labelled nodes is available in order to choose good hyperparameters. In this work, the limitations of the the graph neural network methods are studied and a novel graph-based active learning algorithm is proposed. The second contribution of this thesis is to introduce a modified problem setting that takes into account the timeline of the active learning process. This new formulation leads to a solution that reduces the delay experienced by a labeller interacting with the system. Experiments on node classification benchmark datasets show that the proposed models can outperform state-of-the-art models.

ABRÉGÉ

La classification de nœuds attribués est une tâche importante qui s'inscrit dans multiples applications pratiques. Néanmoins, à l'instar de la majorité des méthodes d'apprentissage, l'utilisation de ces algorithmes nécessite la collecte d'une quantité importante d'étiquettes de données. Ce processus est souvent de longue durée et fastidieux. Afin the résoudre cette problématique, l'apprentissage actif est un domaine de recherche qui tente de limiter le nombre d'étiquettes à collecter tout en maintenant la performance de l'algorithme en sélectionnant les plus informatives. Adopter une approche d'apprentissage actif s'avère alors une avenue prometteuse pour la classification semi-supervisée des nœuds de réseaux attribués. Pour cette raison, cette approche sera le sujet de ce mémoire. Les méthodes de pointe existantes sont basées sur des réseaux de neurones pour données en réseaux, mais leur performance est souvent grandement dépendante des choix des hyperparamètres. Pour utiliser les techniques de sélection d'hyperparamètres, un ensemble important de nœuds étiquetés doit être disponible. Dans ce mémoire, les limites de ces méthodes sont étudiées et un nouvel algorithme d'apprentissage actif pour des données en réseaux est introduit. La deuxième contribution de ce mémoire est d'apporter une modification au problème qui prend mieux en compte la chronologie du processus d'apprentissage actif. Cette nouvelle formulation conduit à une solution qui réduit le délai subi par un oracle interagissant avec le système. Les résultats de tests standard montrent que les modèles proposés surpassent les modèles de pointe.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

CHAPTER 1
Introduction

Traditionally, most learning algorithms have focused on data that belongs to the Euclidean space. While this assumption is not restrictive for most purposes, some applications require more complex data models. One alternative representation is a graph, which can concisely capture the relational structure contained in a network. Graph-structured information cannot be easily transposed to the Euclidean space, but appears in multiple real-world domains such as proteins interactions, telecommunication networks, physical and social interactions, language, and the Internet. As such, it is important to have learning algorithms that are suited to perform inference on this particular type of data.

The development and analysis of these tools are the main goals of the graph learning research community. One of the important tasks within this branch of research is node classification. In this setting, the goal is to classify data points (called nodes) lying on a graph. As an example, a dataset could consist of blog sites that cross reference each other with hyperlinks. Each blog is associated with one of a predefined list of topics, and the node classification algorithm has to learn to infer the topic of a blog site based on its content and the connections between all of the sites. The links give additional information and the graph encodes the relationships between the blogs. Node classification problems arise in various fields and the task has attracted significant interest from the machine learning community. As a result,

multiple algorithms have been proposed [1–7], and they continue to be analyzed and improved upon.

Research into node classification has primarily focused on the semi-supervised setting. In this setting, the algorithm has access to a very limited amount of labelled data. For node classification, this means that labels are provided for only a few nodes. As a result, the proposed solutions for node classification have to heavily rely on the relational information coming from the graph topology. This type of classification is motivated by scenarios where labelled data is tedious to collect. Returning to the previous example of classification of blogs, identifying many websites and collecting their content to form a large dataset can be efficiently performed through automation. However, when it comes to labelling the blogs, assigning each of them to one of the predefined fix set of topics is less straightforward, and generally requires human interaction to achieve extremely high accuracy. If labels can be provided for at least a few examples, then a semi-supervised learning algorithm would be well-suited for the task of classifying the remaining sites.

From the scenario previously described, it is clear that this type of situation can arise in many applications (well beyond the graph related scenarios). Developing solutions that can generalize well without relying on a huge set of labelled data is essential and semi-supervised learning is a good first step toward this goal. Active learning is an approach that strives to optimize the benefits of the labelling process. In active learning, the algorithm is given the additional ability of choosing which points will form the labelled set. Given that we have the opportunity to decide which points to query, we should try to select the most informative points that lead

to the best performance. The active learning process essentially involves alternating between acquiring labels and deciding which label to query next based on what has been learned. Similar to semi-supervised learning, the interest in active learning is motivated by settings in which obtaining labels is particularly challenging or expensive. A compelling application is the medical imaging domain, where the generation of diagnostical labels from medical images or other data requires considerable valuable time from domain experts. As a result, many researchers have explored active learning approaches to medical imaging classification [8–10].

Since most of the node classification research is already directed toward the semi-supervised case, applying active learning seems to be a natural step and is the topic of this thesis. Although there is a rich literature that already tackles active learning for node classification, open problems in this area still exist, especially in the attributed graph case. In an attributed graph, attributes or features are associated with the nodes (and possibly edges) and this information can be exploited by a learning algorithm.

Much of the early research in the field of active learning on graphs focused on graphs with no attributes [11–16]. The common denominator of this line of work is the adoption of probabilistic models that describe the label distribution of the whole graph conditioned on the few available labels. The probabilistic model then forms the foundation for an active learning strategy. Principled methods can be constructed based on a conditional or marginal posterior. The derived techniques are often computationally expensive (and in some cases effectively intractable), so

much effort has been spent on developing approximations that make the computation of quantities needed by the chosen strategy less burdensome. A consequence of the usage of the probabilistic models is that the solutions are intimately tied to the specific graph topology that is analyzed. This can be seen as a positive, since the solution is tailored to each instance, optimized for the graph under study, but it also limits the settings that can be considered. In particular, it means that the algorithms are only effective for the transductive case, where the whole graph is known during the training period. In the transductive case, the proposed algorithms achieve impressive performance. However, as will be highlighted in this thesis, performance dramatically deteriorates when the algorithm is applied in an inductive setting, i.e., when test nodes are added to the graph after training. In addition to this failing, it is challenging to modify most of the algorithms to integrate node attribute information.

More recent work has focused on the attributed graph case [6, 17]. In the algorithms developed for this task, there is interaction between propagation of information across the graph and inference algorithms that learn based on the node features. Thus far, the query selection rules that have been employed are more heuristic in nature and do not rely on probabilistic models [6, 17]. Graph neural networks have recently been established as the state-of-the-art models for node classification, and these emerged also as the model of choice for active learning strategies for attributed graphs. However, the integration of successful deep learning model into an active learning framework should be done with care. It is well known that most neural network algorithms rely on a sizeable validation set of labelled data to properly tune

hyperparameters. If this tuning is not performed, they are often incapable of achieving performance that is competitive with simpler models. In this thesis, we argue that the current state-of-the-art methods based on neural networks have overlooked this and thus operate with the unreasonable and impractical assumption that tuned hyperparameters are available. The implication and impact of this assumption is analyzed in this thesis.

As such, currently there is no graph-based active learning algorithm that can integrate node features, operate with a reasonably limited amount of labelled data, and is applicable to the transductive as well as the inductive case. One of the main contributions of this thesis is to propose a solution based on the Expected Error Minimization framework that meets all of these requirements, while outperforming state-of-the-art methods that have emerged from both branches of graph-based active learning. In addition, the timeline of the traditional active learning formulation is closely examined and based on this, we introduce a modified version of the problem that takes into account the intrinsic delay of the labelling process. This novel formulation leads to a modified version of the initial proposed algorithm, designed to be more time efficient and less frustrating for a human labeller interacting with an active learning system.

1.1 Thesis Organization and Contributions

The organization and contributions of the thesis are summarized below. The material presented in Chapters 4 and 5 of the thesis has been submitted as an 8-page conference paper and is currently under review.

- *Chapter 2 - Background*

  Foundational material associated with the two topics of this thesis (active learning and learning on graphs) is presented in the background chapter. I first present a short summary of basic graph notions and a high level overview of node classification graph learning algorithms. The second part of the chapter is devoted to active learning. First, a formal definition of the active learning problem formulation is given, and this is followed by a summary of the different types of approaches taken in the literature.

- *Chapter 3 - Literature review*

  In this chapter, I review works that have addressed active learning for semi-supervised node classification. The literature review is divided into two parts: one for publications on non-attributed graphs and one for publications on attributed graphs.

- *Chapter 4 - Graph Expected Error Minimization (GEEM)*

  This chapter presents a novel graph-based active learning algorithm for the task of node classification in attributed graphs. The algorithm uses graph cognizant logistic regression for the prediction phase and maximizes the expected error reduction in the query phase. Experiments conducted on four public benchmark datasets demonstrate a significant improvement over state-of-the-art approaches. The development of this algorithm and the implementation of the experiments were my contribution, under the supervision of Prof. Mark Coates.

- *Chapter 5 - Preemptive GEEM*

  This chapter contains the second contribution of this thesis. First I provide motivation for a revised formulation of active learning problem and then I introduce and discuss this formulation. Subsequently, a preemptive querying algorithm that calculates a new query during the labelling process is derived from the previously introduced GEEM algorithm. Experiments conducted on the same four public benchmark datasets demonstrate that the approximations associated with this method have only a minor impact on the performance. Theoretical results that bound the one-step error arising from the approximations are also provided. Again, the development of the algorithm and the implementation of the experiments were my contribution, under the supervision of Prof. Mark Coates. With the assistance of PhD student Soumyasundar Pal, I formulated the theoretical bounds and derived the proofs for the binary classification case,. The proof for the multiclass case was developed in collaboration with Prof. Mark Coates.

- *Chapter 6 - Conclusion*

  This chapter summarizes the main contributions of the thesis and discuss the outcomes and observed results.

CHAPTER 2
Background Material

This chapter provides the key background material for the two main topics of this thesis: prediction on graphs and active learning. In the graph section, the notations and basic concepts of graph theory are first laid out, then a general deep learning model for graph data is presented. The active learning section introduces the general problem formulation of pool-based active learning and gives an overview of the approaches taken by presenting different categories of active learning strategies.

2.1 Learning on Graphs

2.1.1 Graph Theory

A graph is a mathematical representation of a network. In a graph $\mathcal{G}$, entities and their relationships are modeled as sets of nodes and edges: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}, 1 \leq i \leq N$ represents one entity, and the edge $e_{i,j}$, represented as an ordered pair (v_i, v_j), appears in the set of edges if there is a link from node v_i to node v_j. In an *undirected graph*, two nodes are either connected or not; there is no direction to the relationship. This is encoded by always including the two directions in the set of edges $(e_{i,j} \in \mathcal{E}) \iff (e_{j,i} \in \mathcal{E})$. In a *directed graph*, this constraint is not there. The edge $e_{i,j}$ is endowed with a value $a_{i,j}$. In an *unweighted* graph, we always have $a_{i,j} = 1$ for edges that exist. For a *weighted* graph, $a_{i,j}$ can be a real value or *weight* $w_{i,j} \in \mathbb{R}$. In this work, all graphs are considered to be unweighted and undirected unless specified otherwise.

A useful alternative to represent a graph is through its *adjacency matrix* $\mathbf{A}^{N\times N}$. In this square matrix, every possible edge that can exist is represented by an entry in the matrix. This entry is 0 if the link is not present in the graph; if the edge does exist, the entry is a non-zero value. The position in the matrix encodes which nodes are linked. The (i, j)-th entry of the adjacency matrix is thus:

$$\mathbf{A}_{i,j} = \begin{cases} a_{i,j} > 0 \text{ if } e_{i,j} \in \mathcal{E}\,, \\ 0 \text{ else}\,. \end{cases} \tag{2.1}$$

For an unweighted graph $a_{i,j} = 1$ if there is an edge from i to j.

The nodes that are connected to a given node v_i form this node's *neighborhood* (in the simplified case of an undirected graph). In set notation, the neighbors are specified as $\mathcal{N}^*(v_i) = \{v_j \in \mathcal{V}; e_{i,j} \in \mathcal{E}\}$. In the adjacency matrix, the neighborhood is encoded as the position(s) of the non-zero entry(ies) in the vector corresponding to the i-th row of the adjacency matrix $\mathbf{A}_{i,:}$. For an undirected graph, the *degree* of a node refers to the number of neighbors the node has, i.e., $d(v) = |\mathcal{N}^*(v)|$.

2.1.2 Graph Learning Algorithms

Learning Tasks

We use the term *graph learning task* to refer to any inference performed on graph-structured data; a datum can either be a node in a graph or the graph itself. The particularity of graph-based algorithms that sets them apart from other learning algorithms is their ability to process information coming from features and the graph topology jointly. As each node can have a varying number of relationships, and there

is usually no natural ordering of these relationships, models that require a fixed size and ordered input cannot readily be employed.

Graph Neural Networks

A large family of graph neural network (GNN) models adopts the same high level structure to address graph learning tasks. The idea is to successively model "deeper" representations or embeddings of a node with a standard deep learning architecture and incorporate the influence of the graph structure at each layer of the neural network. In the GNNs most relevant to this thesis, the ℓ-th layer representation of a node v is a real-valued vector of dimension $d^{(\ell)}$: $\mathbf{h}_v^{(\ell)} \in \mathbb{R}^{d^{(\ell)}}$. This is obtained from a process that takes as input the representations at the previous layer $\{\mathbf{h}_u^{(\ell-1)}; u \in V\}$ and the topology to propagate information between nodes.

There are other architectures in the literature, e.g., those proposed in [18], that can include additional information in the computation of the node embeddings. For example, an embedding representing a global state of the graph can also contribute to the representation of each node [18]. The earliest GNN models, proposed in [19, 20], also differ from the architectures we present here. They combine recurrent neural networks with message passing across the graph and can accommodate more general combinations of node and graph features. The very high computational burden of these early algorithms restricts their application to small graphs.

Following the framework of [21], the layered computation in most GNNs can be divided into two main steps. The first is the *aggregation* of the information coming from the neighbors into a summary vector $\mathbf{a}_v^{(\ell)}$. The aggregation is performed using a flexible $AGGREGATE$ function:

$$\mathbf{a}_v^{(\ell)} = AGGREGATE\big(\{\mathbf{h}_u^{(\ell-1)} : u \in \mathcal{N}^*(v)\}\big). \tag{2.2}$$

We note that this function operates on an unordered set of vectors. The second step is the *combination* of this neighborhood representation $\mathbf{a}_v^{(\ell)}$ with the previous node representation $\mathbf{h}_v^{(\ell-1)}$. This is performed using a generic $COMBINE$ function:

$$\mathbf{h}_v^{(\ell)} = COMBINE^{(\ell)}\big(\mathbf{h}_v^{(\ell-1)}, \mathbf{a}_v^{(\ell)}\big). \tag{2.3}$$

After these two operations have been defined, $\mathbf{h}_v^{(\ell)}$ can be obtained from the output of the previous layer $\ell-1$ as:

$$\mathbf{h}_v^{(\ell)} = COMBINE^{(\ell)}\big(\mathbf{h}_v^{(\ell-1)}, AGGREGATE\big(\{\mathbf{h}_u^{(\ell-1)} : u \in \mathcal{N}(v)\}\big)\big). \tag{2.4}$$

The $AGGREGATE$ function takes as input an unordered set of vectors, and the cardinality of the set varies according to the size of a node's neighborhood. Since there is no intrinsic way to order neighbors, the function should be permutation invariant. Some possible choices for this function include the weighted sum [2, 4], or sampling followed by concatenation [3]. The $COMBINE$ function usually includes learnable weights and involves a non-linear function $g()$.

A popular GNN is the Graph Convolutional Network (GCN) [2]. This model does not make any distinction between information coming from the neighbors and the information coming from the node itself. It uses a weighted mean operator to combine the vectors in the neighbourhood, then multiplies the result with the learnable weights of its ℓ-th layer: $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d^{(\ell)} \times d^{(\ell-1)}}$. The weights used in the weighted mean operation control the contribution of each neighbor to the output.

Intuitively, these should depend on the graph topology. In a GCN, the weights are derived via a preprocessed adjacency matrix that is constructed by first adding an identity matrix $\mathbf{I}^{N\times N}$ to the adjacency matrix to ensure that all nodes have a self-contribution. Then each node is normalized by its obtained degree. The preprocessed adjacency matrix is thus $\hat{\mathbf{A}} \triangleq \tilde{\mathbf{D}}^{-1/2}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-1/2}$. Here $\tilde{\mathbf{D}}$ is a diagonal matrix that has $d_i + 1$, the degree of node i plus one, as its i-th diagonal entry. The GCN computation at the ℓ-th layer of the architecture can thus be written as:

$$\mathbf{a}_v^{(\ell)} = WEIGHTEDMEAN\big(\mathbf{h}_u^{(\ell-1)} : u \in \{v \cup \mathcal{N}^*(v)\}\big)\,, \tag{2.5}$$

$$\mathbf{h}_v^{(\ell)} = g\Big(\mathbf{W}^{(\ell)}\mathbf{a}_v^{(\ell)}\Big)\,. \tag{2.6}$$

These two operations can be combined into a single expression:

$$\mathbf{h}_v^{(\ell)} = g\Big(\mathbf{W}^{(\ell)} \sum_{u\in\{\{v\}\cup\mathcal{N}^*(v)\}} \hat{a}_{v,u}\mathbf{h}_u^{(\ell-1)}\Big)\,. \tag{2.7}$$

The operation that computes the ℓ-th representation of every node at the same time can be written compactly by concatenating the node embeddings in a matrix $\mathbf{H}^{(\ell)}$ (where $\mathbf{H}_{i,:}^{(\ell)} = \mathbf{h}_{v_i}^{\top(\ell)}$). The computation at each layer is then:

$$\mathbf{H}^{(\ell)} = g\Big(\hat{\mathbf{A}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)\top}\Big)\,. \tag{2.8}$$

In this thesis, we focus on node-level prediction tasks, meaning that we are interested in predicting quantities associated with individual nodes in the graph. However the applications of GNNs go beyond localized inference. They can be applied for graph classification, for example, treating the whole graph as a data point [3, 21–23].

In such a case, the same architecture can be employed with an additional layer that aggregates the information coming from all the nodes to form one global prediction.

2.2 Active Learning

In active learning, an algorithm has the additional ability of choosing which data is labelled. In this context, labeling a point means that an oracle provides its associated output (this can apply for both classification and regression). An illustrative example of active learning given in [24] is the 1-dimensional binary classification problem. Assuming that the data is separable, which points should be selected to find the threshold the quickest?

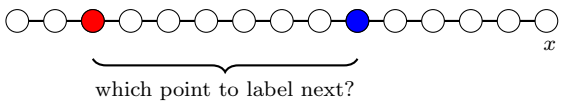


Figure 2–1: Active learning in 1-dimensional binary classification.

In this setting, it is obvious that randomly labeling points is not the sensible thing to do. The optimal active learning strategy is to follow a binary search algorithm to select the points. As the problems grow in complexity and dimensions, the solutions become less trivial. Development of suitable algorithms is the main focus of active learning research.

2.2.1 Active Learning: A Problem Formulation

The data is partitioned into two sets: a small initial labelled set $\mathcal{L}$ from which we have access to the labels $\mathbf{y}_{\mathcal{L}}$ and a set $\mathcal{U}$ consisting of unlabelled points from which we can query the label. The algorithm is restricted to a budget of b nodes that it can query from $\mathcal{U}$ to augment $\mathcal{L}$. The goal is to define a method that can select the best nodes $\mathcal{Q}$ to add to $\mathcal{L}$, in order to optimize the performance on a test set $\mathcal{T}$

throughout the query process. How this test set is formed depends on the nature of the problem. A *pool-based* active learning algorithm alternates between two steps:

1. **Prediction Step**: Form a prediction on $\mathbf{y}_{\mathcal{T}_t}$ based on the current $\mathbf{y}_{\mathcal{L}_t}$ and other information $\mathcal{D}$, $\mathbf{y}_{\mathcal{T}_t} = f_i(\mathcal{D}, \mathbf{y}_{\mathcal{L}_t})$.
2. **Query Step**: Until the budget has been exhausted, select nodes $\mathcal{Q}_t \subset \mathcal{U}_t$ to add to the labelled set. We then update the sets: $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \mathcal{Q}_t$, $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \mathcal{Q}_t$.

The index $t \in \{0, \ldots, b\}$ indicates the current iteration of the active learning process. The test set is also indexed in this problem formulation to cover the case where it coincides with the unlabelled set, but it can also be a fixed set of nodes distinct from $\mathcal{U}$. It is clear that the main step that is unique to active learning is the query step. Most of the contributions in the active learning literature focus on proposing query strategies; these are then coupled with appropriate inference algorithms.

2.2.2 Active Learning Strategies

In [25], Sen et al. define six broad categories for the various query strategies that can be employed: (i) *uncertainty sampling*, (ii) *expected model change*, (iii) *expected error minimization*, (iv) *variance reduction*, (v) *query-by-committee*, and (vi) *density-weighted sampling*.

I now provide a brief summary of the categories. *Uncertainty sampling* aims to query the point about which the predictive algorithm is the most unsure. The intuition behind this strategy is that gaining knowledge about high uncertainty instances will help the model the most and will lead to better performance. As this is a heuristic method, there is no principled way to define the uncertainty measure and multiple metrics have been proposed. A probabilistic approach can be adopted; this

requires us to make an approximation of the prediction's distribution. For classification, some probabilistic quantities that have been proposed are based on 0/1 error, probability margins and entropy. Non-probabilistic methods can also be employed; these are often based on majority-voting or the distance from decision boundaries. In regression, the variance can be used as an uncertainty measure. This category of methods has the drawback of being heuristic. Many of the algorithms can be misled by outliers, but they often perform well in practice and they are computationally efficient.

Similar to the previous category, *Expected Error Change* (EEC) also relies on intuition and its effectiveness is demonstrated through empirical results. In this framework, the most informative query is considered to be the one that will have the biggest impact on the predictive model. There are two components to be considered when forming a metric to predict the impact. First, as the model change depends on what happens after a point is added, we need to form an estimate of this unknown by taking the expectation over the outcomes that can arise for each candidate point. Second, we need to define how to measure the impact of adding a point on a model. Although many metrics have been proposed, a convenient choice is the Euclidean norm of the loss function gradient vector with respect to the candidate point. The main motivation for this choice is that most learning algorithms are gradient-based. The EEC approach is usually more computationally involved than uncertainty sampling as we need to calculate the expectation and compute the gradient at each point over the label space. Even with this overhead, it is generally

considered to be one of the faster query strategies. It also shares with uncertainty sampling the predisposition of being sensitive to outliers.

Another category of methods that considers the state of the model after a potential point has been added is *expected error minimization* (EEM), also called *expected error reduction* (EER). It is a more principled approach as it can directly target the quantity of interest — the generalization error. To do so, the unknown points are treated as a proxy for unseen data and the error on this set of points is approximated by taking the expectation. For each candidate point, we have to obtain a new model, then compute the expected error of this new model over the remaining unlabelled set. This quantity is referred to as the *risk*. As a result, every point in the unlabelled set is involved in each risk computation. For this reason, it is considered to be the most computationally intensive active learning strategy.

Variance reduction has been introduced to address this limitation. The idea is that the targeted error can be reduced indirectly by minimizing the output variance. This stems from the well-known decomposition of the generalization error into three terms: the noise, the bias and the variance. Since a model cannot reduce its own bias, a metric that is based on optimizing the variance is sufficient. This formulation leads to variance-based metrics that can be manipulated to yield solutions of reduced complexity compared to EEM.

Query-by-committee (QBC) is another method that has a solid theoretical foundation. Rather than focusing on the error of the model, the aim is to restrict the number of candidate models once the point is added. Given a labelled set, there is a collection of hypotheses that are consistent with the dataset. The term 'consistent'

means that they do not conflict with any of the points contained in this set, i.e., for classification all of the consistent hypotheses have 100% accuracy. The space spanned by the parameters of these hypotheses is called the version space, and reducing this space helps the training process by making the search for a solution easier. This is the motivation of QBC. A set of hypotheses (or committee members) is maintained and we select the point that maximizes the level of disagreement amongst the members. This disagreement metric can take the form of a distribution distance measure such as the Kullback-Leibler Divergence (KL), Jensen-Shannon, or another entropy-based voting method.

Lastly, in *Density-weighted sampling*, the strategy is to identify and select representative points of the dataset. The underlying assumption is that there are some clusters in the data and knowing some key center points is enough to correctly predict the others. To perform the selection, a similarity metric between a point and the rest of the data needs to be defined. Then we simply select the point that maximizes this metric.

2.3 Summary

Now that foundations of both topics have been presented separately, the following chapter will review how active learning has been applied to the problem of node classification in the existing graph learning literature.

CHAPTER 3
Literature Review

This literature review covers pool-based active learning for node classification. The goal of the node classification task is to predict the label of each node in a test set based on the provided graph topology and any available node features. The literature review provides more detailed description of the algorithms that are more closely related to the methodology proposed in this thesis. Early research on this topic primarily focused on non-attributed graphs [11–16]. The inherent structure in graph data alone allows us to postulate powerful models that do not rely on node features. As a result, active learning for models that do not process or require node attributes was explored. In Section 3.1 we review models for learning on non-attributed graphs and in Section 3.2 we review active learning algorithms for such graphs. Section 3.3 presents the active learning algorithms for attributed graphs that have been proposed in the research literature.

3.1 Non-Attributed Graph Models

The graph models presented in this section build on the intuition that if two nodes are connected, then they are more likely to share the same label. This intuitive and simple assumption can be expressed with a Binary Random Markov Field (BRMF) (here we only consider the binary case for simplicity, but most algorithms can be extended to the multi-label case through a one-vs-rest scheme). In a BRMF, each edge that links nodes of dissimilar labels $y_i \neq y_j$ makes a negative contribution

in the likelihood. Given a strength parameter β and a normalizing constant Z, the probability of observing labels $\mathbf{y}$ on a graph with adjacency matrix $\mathbf{A}$ under this model can be written as :

$$\mathbb{P}(\mathbf{y}) = \frac{1}{Z} \exp\left(-\frac{\beta}{2} \sum_{i<j}^{N} a_{ij} \left(y_i - y_j \right)^2 \right). \tag{3.1}$$

From this formulation, we can define a posterior of unknown labels conditioned on the labelled nodes (and the graph topology implicitly). The issue is that evaluating this posterior is a combinatorial problem. For tractability, researchers have introduced relaxations and approximation strategies.

As a side note, it is clear that this model is completely specified by the graph through the $a_{i,j}$ values that encode the presence of edges. This restricts the applicability of the model to the graph used for training. For this reason, non-attributed models focus on the transductive setting, where the entire graph is known, both during training and testing. In the transductive setting, the test set corresponds to the unlabelled set, hence all nodes are contained in $\mathcal{U} \cup \mathcal{L}$.

In [26], the BMRF is relaxed to a continuous formulation. Moving away from the discrete domain usually comes with a decrease of complexity, and in this case, it significantly diminishes from combinatorial to linear. Rather than targeting discrete labels $y \in \{0, 1\}$, Zhu et al. define in [26] a real valued function f that is constrained to take the true values at the labelled nodes $f(\ell) = y_\ell, \ell \in \mathcal{L}$. This leads to a Gaussian Random Field (GRF) model:

$$\mathbb{P}(\mathbf{y}) = \frac{1}{Z} \exp\left(-\frac{\beta}{2} \sum_{i<j}^{N} a_{ij} \left(f(i) - f(j) \right)^2 \right). \tag{3.2}$$

The energy function associated with this model is $E(f) = \frac{1}{2}\sum_{i<j}^{N} a_{ij}\left(f(i) - f(j)\right)^2$. It is easily shown that the function f that minimizes this energy function, subject to the constraint $f(\ell) = y_\ell, \ell \in \mathcal{L}$, is *harmonic*. That is, it satisfies the property $\mathbf{\Delta} f = 0$ on unlabelled points, where $\mathbf{\Delta}$ is the combinatorial Laplacian, which can be defined in matrix form as $\mathbf{\Delta} = \mathbf{D} - \mathbf{A}$. This harmonic property allows Zhu et al. to pose the equality:

$$\mathbf{f}_\mathcal{V} = \mathbf{D}^{-1}\mathbf{A}\mathbf{f}_\mathcal{V}. \tag{3.3}$$

In this expression, $\mathbf{f}_\mathcal{V}$ is an explicit vector of the values of f at each node. By decomposing $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{f}$ in block matrices for labelled nodes and unlabelled nodes as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\mathcal{L},\mathcal{L}} & \mathbf{A}_{\mathcal{L},\mathcal{U}} \\ \mathbf{A}_{\mathcal{U},\mathcal{L}} & \mathbf{A}_{\mathcal{U},\mathcal{U}} \end{bmatrix}, \tag{3.4}$$

the solution for the unlabelled nodes $\mathbf{f}_\mathcal{U}$ can be found in closed form:

$$\mathbf{f}_\mathcal{U} = -\mathbf{\Delta}_{\mathcal{U},\mathcal{U}}^{-1}\mathbf{\Delta}_{\mathcal{U},\mathcal{L}}\mathbf{f}_\mathcal{L}. \tag{3.5}$$

All that remains is to translate the real values in $\mathbf{f}_\mathcal{U}$ back to binary labels by thresholding: $\{\hat{y}_i = 0; \mathbf{f}_i \leq 0.5 \wedge i \in \mathcal{U}\}$.

3.2 Active Learning for Non-Attributed Graphs

GRF-EEM. After proposing the Gaussian random field model for non-attributed graphs in [26], Zhu et al. published a second article [11], in which they employed the tractable probabilistic model to tackle the problem of active learning. Under the proposed model, the probability distribution of the real-valued labels of the unlabelled nodes, conditioned on the observed labels, is a multivariate normal distribution with

a mean of $\mathbf{f}_{\mathcal{U}}$ and a variance that depends on the Laplacian. The conditional distribution can be approximated as $\mathbb{P}(\mathbf{y}_{\mathcal{U}}|\mathbf{y}_{\mathcal{L}}) \approx \mathcal{N}(\mathbf{f}_{\mathcal{U}}, \mathbf{\Delta}^{-1}_{\mathcal{U},\mathcal{U}})$, and this can be used to determine the best query using the EEM active learning framework.

As mentioned in Chapter 2, a key quantity that needs to be defined for EEM active learning is the risk associated with a model. The true Bayes risk for the node classification task is:

$$R_{Bayes}(f) = \sum_{i=1}^{N} \sum_{y \in \{0,1\}} \mathbb{1}[\hat{y}_i \neq y] \mathbb{P}(y_i = y|\mathbf{y}_{\mathcal{L}}) \tag{3.6}$$

Using the multivariate Gaussian approximation of the true conditional distribution, Zhu et al. derive the following approximate risk $R(f)$:

$$R(f) = \sum_{i=1}^{N} \min\left(f(i), 1 - f(i)\right). \tag{3.7}$$

Following the EEM framework, the node to query is the node that leads to the largest expected reduction of this risk once it is added to the labelled set. The risk is computed for each candidate node $q \in \mathcal{U}$ and is denoted by R^{+q}. As we do not have the label of the candidate node before executing the query, to obtain the risk we need to: 1) take the expectation with respect to the candidate node label y_q, and 2) recompute the model as if the candidate node had ground truth label y_q, denoted as $f^{+(q,y_q)}$. The risk is then

$$\begin{aligned} R^{+q} &= E_{y_q}[R(f^{+(q,y_q)})] && (3.8)\\ &= \mathbb{P}(y_q = 0|\mathbf{y}_{\mathcal{L}}) R\left(f^{+(q,0)}\right) + \mathbb{P}(y_q = 1|\mathbf{y}_{\mathcal{L}}) R\left(f^{+(q,1)}\right) && (3.9)\\ &= (1 - f(q)) R\left(f^{+(q,0)}\right) + f(q) R\left(f^{+(q,1)}\right) && (3.10) \end{aligned}$$

Recomputing f for every candidate node q appears at first to be computationally expensive, but Zhu et al. derive an effective method through the following formulation:

$$\mathbf{f}_{\mathcal{U}}^{+(q,y_q)} = \mathbf{f}_{\mathcal{U}} + (y_q - f_q)\frac{(\mathbf{\Delta}_{\mathcal{U},\mathcal{U}}^{-1})_{:,q}}{(\mathbf{\Delta}_{\mathcal{U},\mathcal{U}}^{-1})_{q,q}} \tag{3.11}$$

This allows the development of an algorithm of order $O(N^2)$, which is not computationally prohibitive and can be used in practical settings.

V-optimality and Σ-optimality. As presented in Chapter 2, one strategy to avoid the complexity of *Expected Error Minimization* is to target other proxy quantities. In [15], Ji and Han build on the method proposed in [11] by proposing the variance reduction criterion, V-optimality. The developed algorithm targets the trace of the variance of the GRF: $\mathrm{Tr}\left(\mathbf{\Delta}_{\mathcal{U},\mathcal{U}}^{-1}\right)$. An alternative variance reduction approach is proposed in [16]. The implicit loss used in V-optimality is the L^2 regression loss:

$$L^2 = \sum_{u \in \mathcal{U}} \left(y_u - f(u)\right)^2 . \tag{3.12}$$

In [16], Ma et al. argue that a survey loss provides a better approximation of the true 0/1 error. The survey error L_{survey} is the discrepancy in the proportion of nodes belonging to a class:

$$L_{survey} = \left(\mathbf{1}^\top \mathbf{y}_{\mathcal{U}} - \mathbf{1}^\top \mathbf{f}_{\mathcal{U}}\right)^2 . \tag{3.13}$$

Interestingly, the labels are absent from the two loss expressions. The selected node q^* only depends on the unlabelled set, with the candidate node q removed, $\mathcal{U}^{-q} \triangleq \mathcal{U} \backslash \{q\}$:

$$\text{V-optimality: } q^* = \arg\min_{q \in \mathcal{U}} \operatorname{Tr}\left(\mathbf{\Delta}^{-1}_{\mathcal{U}^{-q},\mathcal{U}^{-q}}\right). \tag{3.14}$$

$$\Sigma\text{-optimality: } q^* = \arg\min_{q \in \mathcal{U}} \mathbf{1}^\top \mathbf{\Delta}^{-1}_{\mathcal{U}^{-q}} \mathbf{1}. \tag{3.15}$$

Ma et al. derive the optimal order to follow to add nodes sequentially in [16], and they also present Σ-optimality as being advantageous on two fronts. First, unlike alternative query selection methods that need to add the nodes one-by-one because the computation at each step is dependent on the previous point that has been added, this formulation allows selection of a batch of nodes. This gives the advantage of targeting a global solution rather than greedily minimizing the expected error at each step. In the experiments in [16], the method outperforms the GRF-EEM method of [11] and other uncertainty sampling and clustering-based baselines. Second, Σ-optimality also avoids the tedious task of retraining at each added point.

In the remainder of the thesis, we refer to solutions like these, in which selection is not impacted by the labels of the queried nodes, as *non-adaptive* methods. Although there are some advantages, ignoring the labels of the nodes can reduce the effectiveness of the learning process. Intuitively, such methods appear to ignore valuable information.

Mincut algorithms. Although BMRF-based methods are predominant in the active learning literature for non-attributed graphs, other approaches also exist. Several methods are based on graph theory, and we now summarize the most closely related approaches.

Binary node classification can be framed as a clustering or a minimum cut problem. A cut is a partitioning of the nodes into two sets, and $\Phi(\mathbf{y})$ is defined as being the sum of the weights of edges that cross the two partitions of a cut:

$$\Phi(\mathbf{y}) = \sum_{i<j}^{N} a_{ij}|y_i - y_j|. \tag{3.16}$$

This value is also called the smoothness coefficient. This equation is obviously very similar to the BMRF; although it is expressed differently, the underlying assumption is the same. The goal is to choose $\mathbf{y}_{\mathcal{U}}$ such that $\Phi(\mathbf{y})$ is minimized.

In [27], the goal is to sample the initial set *offline* that will be subsequently used to infer the labels, which can be seen as querying a batch of nodes only once, i.e., addressing the case $|\mathcal{Q}_0| = b$ and $\mathcal{L}_0 = \varnothing$. The label selection method relies on optimizing the graph cut induced by the selection. Guillory and Bilmes additionally developed bounds that relate the smoothness coefficient of a graph to the classification error of the solution [27].

Results from [27] were subsequently leveraged to build an active learning framework suited for the special case of trees [12, 13]. Cesa-Bianchi et al. proposed an extension to general graphs by using spanning trees but they did not provide theoretical guarantees on the optimality of their solution for the more general case [13]. This open problem was addressed in [28], where a submodular function analysis was used to derive a bound on the quality of the approximation for the case of general graphs.

Error bound minimization. In [14], Gu et al. propose an alternative active learning approach for unattributed graphs, called Local and Global Consistency

(LLGC), that exhibits performance that is competitive with the Mincut and GRF methods. For the proposed LLGC model, Gu et al. derive an upper bound on the expected error of label prediction on the unlabelled set, and show that this can be decomposed into the empirical error on the labelled data plus the empirical transductive Rademacher complexity of LLGC. Since the empirical error is fixed, the strategy is to minimize the Rademacher complexity. This is similar to the variance reduction idea.

Two-step Approximation (TSA). In [29], Jun and Nowak take a different approach to EEM by moving away from the Gaussian random field model. Rather than relaxing the problem to a continuous formulation, the BRMF model is retained and its combinatorial complexity is addressed via approximation. Before detailing the solution, we elaborate upon the complexity of the problem of using BMRF with EEM. We can restate the BMRF model using the introduced Laplacian matrix:

$$\mathbb{P}(\mathbf{y}) = \frac{1}{Z} \exp\left(-\frac{\beta}{2}\mathbf{y}^\top \mathbf{\Delta} \mathbf{y}\right) . \tag{3.17}$$

To compute the posterior marginal, $\mathbb{P}\left(y_i|\mathbf{y}_\mathcal{L}\right)$, of the label of a node i given $\mathbf{y}_\mathcal{L}$, each permutation of the marginalized node labels has to be considered. This is where the combinatorial nature of the problem arises. In developing an approximation, as in GRF-EEM, the true Bayes risk is used as a starting point. In this case, it leads to a risk, $R\left(y_q = y, \mathbf{y}_\mathcal{L}\right)$, defined as a function of the given labels:

$$R\left(y_q = y, \mathbf{y}_\mathcal{L}\right) = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \max_{y' \in \{1,-1\}} \mathbb{P}\left(y_i = y' | y_q = y, \mathbf{y}_\mathcal{L}\right)\right) . \tag{3.18}$$

So the risk of adding a node $R^{+q} = E_{y_q}[R(y_q = y, \mathbf{y}_\mathcal{L}))]$ is:

$$R^{+q} = \mathbb{P}(y_q = -1|\mathbf{y}_\mathcal{L})R(y_q = -1, \mathbf{y}_\mathcal{L}) + \mathbb{P}(y_q = 1|\mathbf{y}_\mathcal{L})R(y_q = 1, \mathbf{y}_\mathcal{L}). \tag{3.19}$$

This formulation highlights the importance of developing an effective method to compute a conditional label probability $\mathbb{P}(y|\mathbf{y}_\mathcal{L})$; the minimization of risk requires this probability to be evaluated many times.

To address this, Jun and Nowak introduce a Two-Step Approximation (TSA) of this posterior marginal [29]. First they use the log probability ratio approximation to approximate the conditional using some function $\mu()$:

$$\mathbb{P}(y_q = 1|\mathbf{y}_\mathcal{L}) \approx \sigma\left(\log \mu(y_q = 1, \mathbf{y}_\mathcal{L}) - \log \mu(y_q = -1, \mathbf{y}_\mathcal{L})\right). \tag{3.20}$$

The next step is to find a suitable μ that can act as an upper bound on the joint log probability, i.e., $\log(\mathbb{P}(y_q, \mathbf{y}_\mathcal{L})) \leq \log(\mu(y_q, \mathbf{y}_\mathcal{L}))$. The two approximations involve using a max operator instead of log-sum-exp and relaxing the resultant integer maximization problem. The final derived approximation is:

$$\mathbb{P}(y_q = 1|\mathbf{y}_\mathcal{L}) \approx \sigma\left(-2\boldsymbol{\Delta}_{q,\mathcal{L}}\mathbf{y}_\mathcal{L} + 2\boldsymbol{\Delta}_{\mathcal{U}^{-q},q}\boldsymbol{\Delta}^{-1}_{\mathcal{U}^{-q},\mathcal{U}^{-q}}\boldsymbol{\Delta}_{\mathcal{U}^{-q},\mathcal{L}}\mathbf{y}_\mathcal{L}\right), \tag{3.21}$$

where $\boldsymbol{\Delta}_{r,s}$ denotes the matrix or vector obtained by taking the r-th row(s) and s-th column(s) of the laplacian matrix as it was used before. Jun and Nowak also develop a way to reduce the computational overhead to obtain a complexity of $O(N^2)$, which is of the same order as the other GRF-based methods. In their experiments on real data, they show that TSA either outperforms or is on-par with the previously introduced GRF-EEM, V-optimality and Σ-optimality algorithms. Empirical results in [29] also

highlight how non-adaptive methods lead to a poorer exploitation/exploration trade-off than adaptive methods such as TSA.

Gaussian Random Field – Expected Change (GRF-EC). [30] is the most recent publication in the line of work that uses Gaussian random fields for active learning in graphs. Berberidis et al. apply the *Expected Change* strategy on a slightly modified version of the GRF. The authors point out that the initial GRF model had a bias toward the zero class as it was using 0/1 labels, $y \in \{0, 1\}$, while using a zero-mean Gaussian field. A simple fix is to use different labels, $y \in \{-1, 1\}$.

As the Expected Change strategy is to select the node that leads to the most change, the change of a model conditioned on different labelled sets needs to be quantified. Berberidis et al. propose five different expected change (EC) metrics and conduct a thorough analysis of the relative advantages and disadvantages of each EC metric, compared to each other and to the competitive baselines (GRF-EEM, V-optimality, Σ-optimality, TSA). The metrics are 1) *FL*, a count of the number of flipped labels; 2) *KLG*, the Kullback–Leibler (KL) divergence of the distributions of the continuous random variables of the unlabelled nodes, modeled by the GRF; 3) *KL*, the KL divergence of the discrete random labels of the unlabelled nodes, modeled as Bernoulli variables; 4) *MSD*, the mean-square deviation of the GRF models; and 5) *TV*, the total variation of the GRF models. An uncertainty-based correction that can be added to TV and MSD is also proposed. These two metrics combined with this augmentation end up being the most promising candidates, offering advantages both in terms of computational effectiveness and in performance.

Discussion. Common denominators of this branch of research are well-defined models that can lend themselves to theoretical analysis and more principled active learning strategies; namely EEM and VR. Although [30] employs EC, the authors were able to include mathematical connections with uncertainty sampling, V-optimality, and Σ-optimality. It is notable that committee-based strategies, which are prevalent throughout the active learning literature, have been largely ignored in the literature on active learning for unattributed graphs. A possible explanation is that it is more challenging to derive a diverse committee, because without attributes, all algorithms are based on similar models parameterized by the observed graph. Forming a committee of coherent hypotheses that disagree with each other at some unlabelled points can be challenging.

Although the strategies presented in this section offer the advantage of being principled methods that directly target the quantity we want to optimize, label propagation-based models cannot take into account node features and consequently must rely on strong assumptions regarding the relationships between the graph topology and the data. Most label propagation methods struggle if the graph is not connected and do not usually translate well to an inductive setting, because the model and hence the query decisions heavily rely on the knowledge of the complete graph topology.

3.3 Active Learning for Attributed Graphs

In contrast to the non-attributed graph active learning literature, works that consider the attributed graph setting are more prone to employ mixtures of heuristics and are not tied to one predictive model.

Active Learning For Networked data (ALFNET) [31], introduced by Bilgic et al., is arguably the first work to address active learning for attributed graphs. It was preceded by work by the same authors [32, 33] that tackled the related problem of label acquisition for collective classification (another name for node classification). In this earlier work, the acquisition cost (or query cost) is explicitly modeled in the loss function and jointly minimized at inference time with respect to a batch of selected nodes. Another departure of the label acquisition framework from the active learning formulation is that the inference algorithm is assumed to be already trained and provided. The algorithm in [32, 33] is based on avoiding labeling nodes that would induce misclassification, and the selection is made by targeting nodes that minimize the potential error.

ALFNET was introduced after GRF-EEM [11] and before V-optimality [16]. The task addressed is inductive node classification. As discussed, algorithms developed for unattributed graphs do not extend well to this setting. Relative to GRF-EEM, Bilgic et al. position their contribution as being free of strong model assumptions and emphasize the significant time complexity of EEM frameworks.

The predictive model employed in ALFNET is the Iterative Classification Algorithm (ICA) [34]. The idea behind ICA is to form a node embedding by aggregating the information coming from the node's features, but also from the label and the features of the neighbors. Since the majority of the nodes will mostly have unlabelled neighbors, the algorithm alternates between making intermediate predictions and, from those predictions, augmenting the label set by retaining some of the predictions

as ground truth. ALFNET uses two classifiers, one that takes into account the network and one that only considers features. With these classifiers, the query strategy is a mixture of uncertainty sampling, committee sampling and density sampling. A batch of nodes is selected at each iteration.

Active learning for Graph Embedding (AGE). [17] followed the recent introduction of the Graph Convolution Network [2]. Motivated by the success of graph neural networks, Cai et al. leveraged the output of a GCN to design active learning metrics. The process slightly differs from the framework presented in earlier sections, because the separation between prediction and query steps is blurred. Instead of fully training the predictive model at each active learning step, the algorithm only trains for one epoch before adding a query. Training until convergence is only performed once the given budget has been reached.

The query strategy is based on a score that is a weighted mixture of three metrics covering different active learning strategies. These include an uncertainty metric and two density-based metrics. The uncertainty metric is obtained by calculating the entropy of the softmax output given by the current GCN model. The softmax output of an incomplete trained GCN is viewed as an approximation of the label distribution. The GCN output is also used to provide node embeddings for a density metric. The embeddings are clustered and the distance computed between each node's embedding and the center of its cluster. A more central embedding indicates a more representative node. The other density metric is based on graph centrality. This metric is independent of the GCN and only relies on the position of the node in the graph.

The quality of the GCN output is expected to be poor in early stages and to improve as training continues. For this reason, the weight that each metric has in the final decision evolves as more nodes are added to the labelled set in order to reflect the increased confidence in the two metrics that are derived from the output of the GCN. The weights are updated following a fixed rule, with a rate parameter that is found by cross-validation.

Multi-armed bandit ARNMAB. The AGE algorithm is extended in [6], where Gao et al. argue that using a fixed weight adaptation rule is suboptimal. Their proposal is to use an alternative multi-armed bandit (MAB) algorithm that learns how to dynamically balance the contributions of the different metrics. They argue that this mechanism can better adapt to the varying natures of different datasets.

Discussion. In this thesis, AGE and ARNMAB will be referred to as GCN-based active learning algorithms. They represent the state-of-the-art methods for active learning on attributed graphs. It is likely that this status can be partly attributed to the powerful generalization capabilities of the GCN. This brings up a point that was touched upon in the introduction. In both GCN-based publications mentioned, the presented active learning experiments start off with an initial labelled set $\mathcal{L}_0$ varying between 10 and 30 nodes depending on the dataset, while using 500 additional labelled nodes as a validation set for tuning hyperparameters and performing early stopping. This is not a sensible or practical setting for an active learning task. It is possible that the prediction quality of the GCN is actually not too sensitive to hyperparameter choices or early stopping. The algorithms do incorporate a mechanism that assumes that the quality of the GCN predictions is poorer at the

beginning. If hyperparameter tuning is not important, then the end performance would not be impacted if no validation set is available. We will see in the following chapter that this is, in fact, a major issue.

3.4 Summary

This literature review chapter presented the works relevant to the topic of this thesis. Two directions of research have been identified and the different types of solutions each tend to generate have been described. Even though they tackle a slightly different problem setting, the state-of-the-art methods for non-attributed graphs can often achieve similar performance to the GCN-based methods. The models designed for the non-attributed case have the disadvantage of not being able to use feature information, but their model formulation allows them to be applied even when there are very few labels. For these reasons, they are included as baseline in this thesis in order to give a more complete picture of active learning on graphs. Based on the review in this chapter, state-of-the-art methods can be identified for both fields; TSA and EC for the non-attributed graph case and AGE and ARNMAB for attributed graphs (subject to the hyperparameter tuning concern). In addition, the limitations and disadvantages of both approaches have been identified and discussed. This has allowed us to clearly define what is currently lacking from the literature in this field and which characteristics a solution should have in order to make a meaningful contribution. In the following chapter, such a solution is proposed.

CHAPTER 4
Graph Expected Error Minimization (GEEM)

In this chapter, we propose a novel EEM-based algorithm for active learning on attributed graphs. The chapter commences with the problem formulation and specifies the scope of the work. Then I present the methodology of the core algorithm. This is followed by a second version which is equipped with a mechanism to cover additional active learning scenarios. Section 4.4 presents the experimental details and results for three different numerical experiments. The performance is compared with state-of-the-art algorithms for active learning on attributed graphs. The chapter concludes with a discussion of the results.

4.1 Problem Setting

The primary contributions of this thesis are novel methods for pool-based active learning for the transductive and inductive semi-supervised node classification task on attributed graphs. In both classification settings, there is an attributed graph $\mathcal{G}$, node features of dimension $d^{(0)}$ contained in the matrix $\mathbf{X} \in \mathbb{R}^{N \times d^{(0)}}$ and node labels $\mathbf{Y}$ (the labels are now expressed in a one-hot encoding matrix). To complete the general active learning problem formulation presented in Section 2.2, we need to specify exactly how $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{T}$ are formed. The partitioning depends on the experiment setting, and details are provided in Section 4.4. The remainder of the problem formulation is the same as in Section 2.2; a small initial labelled set $\mathcal{L}_0$

($|\mathcal{L}_0| \ll |\mathcal{V}|$) is given with true node labels $\mathbf{Y}_{\mathcal{L}_0}$, to which b nodes can be added from the unlabelled set $\mathcal{U}_0$. Evaluation is performed on the nodes of the test set.

1. **Prediction Step**: Inference depends on the type of classification. For the transductive setting, the full feature matrix $\mathbf{X}$ and graph topology $\mathcal{G}$ are available in addition to $\mathbf{Y}_{\mathcal{L}}$ so the prediction is given by: $\hat{\mathbf{Y}}_{\mathcal{T}} = f_t(\mathbf{X}, \mathcal{G}, \mathbf{Y}_{\mathcal{L}_t})$. For the inductive setting, we do not have access to the node features of the test nodes: $\hat{\mathbf{Y}}_{\mathcal{T}} = f_t(\mathbf{X}_{\mathcal{V}\setminus\mathcal{T}}, \mathcal{G}_{\mathcal{V}\setminus\mathcal{T}}, \mathbf{Y}_{\mathcal{L}_t})$.
2. **Query Step**: The optimal node is chosen $q_t^* \in \mathcal{U}_t$ and the sets are updated $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{q_t^*\}$, $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{q_t^*\}$.

4.2 Methodology : GEEM

The proposed Graph EEM (GEEM) algorithm is based on the EEM framework and uses a node classification logistic regression model. These two components are first presented separately, and then I describe how they are combined to form the active learning solution.

Expected Error Minimization (EEM). Similar to most of the EEM-based work that was presented in the literature review in Chapter 3, the risk $R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}}$ of adding node $q \in \mathcal{U}_t$ given the current known label set $\mathcal{L}_t$ is derived using the zero-one error. When considering the addition of node q, we denote by $\mathcal{U}_t^{-q}$ the set of unlabelled nodes after t iterations of active learning, excluding node q. The risk associated with adding node q as a query node is then $R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}}$ and is defined as:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} \triangleq E_{y_q}\Big[E_{\mathbf{Y}_{\mathcal{U}_t^{-q}}}\big[\tfrac{1}{|\mathcal{U}_t^{-q}|}\sum_{i\in\mathcal{U}_t^{-q}} \mathbb{1}[\hat{y}_i \neq y_i | y_q, \mathbf{Y}_{\mathcal{L}_t}]\big]\Big]. \tag{4.1}$$

Here $\hat{y}_i$ is the label prediction at node i. The expected error is thus calculated by summing error probabilities over the unlabelled set, excluding the node q that is being considered. Define $\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \triangleq \Big(1 - \max_{k' \in \mathcal{K}} \mathbb{P}(y_i = k' | \mathbf{Y}_{\mathcal{L}_t}, y_q = k)\Big)$, where $\mathcal{K}$ is the set of classes. If the query node y_q has label k, then $\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}}$ represents the probability of making an error in the prediction $\hat{y}_i$ of the label of node i. If we can compute the distribution $\mathbb{P}(y|\cdot)$, we can evaluate the risk of querying q:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} = \tfrac{1}{|\mathcal{U}_t^{-q}|} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \mathbb{P}(y_q = k | \mathbf{Y}_{\mathcal{L}_t}) \,. \tag{4.2}$$

The query algorithm selects the risk-minimizing node q_t^*:

$$q_t^* = \arg\min_{q \in \mathcal{U}_t} R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} \,. \tag{4.3}$$

It remains to define the probabilistic model $\mathbb{P}(y|\cdot)$.

Graph-cognizant logistic regression. Our proposed solution is to use a graph-cognizant logistic regression model to obtain $\mathbb{P}(y|\cdot)$. Such a model was introduced in [35], where it was derived as a simplified (linearized) version of the GCN of [2]. [35] showed that the simplified model can achieve competitive performance for most datasets for a significantly lower computational cost. As highlighted in the publications involving EEM, a new model needs to be learned for every potential query node, so it is essential that the computational cost of model inference is relatively low. The graph-cognizant logistic regression model meets our requirements: its computational requirements are moderate and it takes into account the graph structure and node features.

A GCN is built by stacking layers, as defined in equation 2.8, which we restate here for convenience:

$$\mathbf{H}^{(\ell)} = g\Big(\hat{\mathbf{A}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)\top}\Big)\,.$$

The features of the nodes are the initial node representation: $\mathbf{H}^{(0)} = \mathbf{X}$ and the final layer, paired with a softmax operator σ as the final non-linearity, gives the predicted output: $\mathbf{H}^{(\ell)} = \hat{\mathbf{Y}}$. A GCN with an arbitrary number of intermediate layers ℓ can be expressed as:

$$\hat{\mathbf{Y}} = \sigma\Big(\hat{\mathbf{A}}g\Big(\cdots g\Big(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{\top(0)}\Big)\cdots\Big)\mathbf{W}^{\top(\ell)}\Big). \tag{4.4}$$

To obtain a linearized version, we simply need to remove the non-linear operator from the intermediate layers.

$$\hat{\mathbf{Y}} = \sigma\Big(\hat{\mathbf{A}}\Big(\cdots\Big(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{\top(0)}\Big)\cdots\Big)\mathbf{W}^{\top(\ell)}\Big) \tag{4.5}$$

This can be simplified by an adjacency matrix raised to the ℓ-th power with a single weight matrix $\mathbf{W} \in \mathbb{R}^{d^{(0)}\times d^{(\ell)}}$ to form the final model $\hat{\mathbf{Y}} = \sigma(\hat{\mathbf{A}}^{\ell}\mathbf{X}\mathbf{W})$. By defining $\tilde{\mathbf{X}} \triangleq \hat{\mathbf{A}}^{\ell}\mathbf{X}$, which can be interpreted as graph-based preprocessing of the node features, the final model matches a logistic regression formulation:

$$\hat{\mathbf{Y}} = \sigma(\tilde{\mathbf{X}}\mathbf{W})\,. \tag{4.6}$$

The parameter ℓ controls the number of hops that are considered when generating the final node representation. For most datasets that have been analyzed in the literature, using a 2-hop ($\ell = 2$) neighborhood yields good classification performance.

Graph EEM (GEEM). Using the graph-cognizant logistic regression model, we can compute a risk for each query node. At each step t, we use the current known

labels $\mathbf{Y}_{\mathcal{L}_t}$ to find the weights $\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}}$ by minimizing the error for the predictions $\hat{\mathbf{Y}}_{\mathcal{L}_t} = \sigma(\tilde{\mathbf{X}}_{\mathcal{L}_t}\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}})$. This can be used to approximate $\mathbb{P}(y_q = k|\mathbf{Y}_{\mathcal{L}_t}) \approx \sigma(\tilde{\mathbf{x}}_q\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}})^{(k)}$, where the index (k) indicates that we extract the k-th element of the vector. Then for each candidate node q, for each possible class k, we solve:

$$\hat{\mathbf{Y}}_{\mathcal{L}_t,+q,y_k} = \sigma(\tilde{\mathbf{X}}_{\mathcal{L}_t,+q,y_k}\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t},+q,y_k})\,. \tag{4.7}$$

Here the notation $+q, y_k$ indicates that we are adding node q to the labelled set and assigning it label y_k. For the adopted model, we have

$$\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} = (1 - \max_{k' \in \mathcal{K}} \quad \sigma(\tilde{\mathbf{x}}_i\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t},+q,y_k})^{(k')}). \tag{4.8}$$

The node to query is then the one that minimizes the risk:

$$q_t^* = \arg\min_{q \in \mathcal{U}_t} \tfrac{1}{|\mathcal{U}_t^{-q}|} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \sigma(\tilde{\mathbf{x}}_q\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}})^{(k)}\,. \tag{4.9}$$

This implies that we have a computational complexity of $O(|\mathcal{U}||\mathcal{K}|T)$, where T represents the complexity associated with training the model. For logistic regression, this is the overhead involved in learning the weights $\mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}}$. The solution in our experiments is obtained by using the liblinear solver, a standard iterative algorithm from the scikit-learn library with default parameters and L^2 loss. As a result, the proposed algorithm requires the choice of very few hyperparameters (only the number of hops ℓ and logistic regression hyperparameters). This contrasts with the active learning approaches based on graph neural networks, where there are multiple hyperparameters that must be selected, and suboptimal choices can have a major impact on performance (as demonstrated in the experiments in Section 4.4).

4.3 Methodology : Combined Method

Most of the active learning methods for non-attributed graphs that were presented in Chapter 3 can operate in the most extreme case of active learning where we start with only one labelled node. In this scenario, the logistic regression model cannot make useful predictions until at least some nodes have been queried. To address this problem, the GEEM algorithm can be combined with a label-propagation method. The aim is to first use label-propagation when very few node labels are available, then switch to a combination of both algorithms when more information is available, and finally transition to the more accurate graph-cognizant logistic regression. Bayesian model averaging provides a mechanism to make this transition [36,37]. In Bayesian model averaging, we have M different classifiers and our belief is that one of these models is correct. We start with a prior $\mathbb{P}(m)$ over each model. After observing data $\mathcal{D}$, we compute the model evidence $\mathbb{P}(\mathcal{D}|m)$ and update the posterior using Bayes' rule $\mathbb{P}(v|\mathcal{D}) = \mathbb{P}(v)\mathbb{P}(\mathcal{D}|m)/\mathbb{P}(\mathcal{D})$ to compute and weight the predictions:

$$\mathbb{P}(y|\mathcal{D}) = \sum_{m=1}^{M} \mathbb{P}(y, m|\mathcal{D}) = \sum_{m=1}^{M} \mathbb{P}(y|m, \mathcal{D})\mathbb{P}(m|\mathcal{D}). \tag{4.10}$$

In the context of active learning using expected error minimization, we need to evaluate the risk associated with a query. This risk depends on which probabilistic model is employed, so the model m is added in the notation to indicated which model is being used: $R^{+q}_{|\mathbf{Y}_\mathcal{L},m}$. In the combined method, a model-averaged risk can be computed :

$$R^{+q}_{|\mathbf{Y}_\mathcal{L}} = \sum_{m=1}^{M} R^{+q}_{|\mathbf{Y}_\mathcal{L},m} \mathbb{P}(m|\mathbf{Y}_\mathcal{L})\,. \tag{4.11}$$

In order to compute this expression, we need to evaluate $\mathbb{P}(m|\mathbf{Y}_{\mathcal{L}})$. Assuming that we have equal prior belief in the models available to us, this is equivalent to calculating the marginal likelihood $\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m)$.

Two models are incorporated in our proposed method, one based on label propagation and the other based on logistic regression. For the binary random field model that underpins the label propagation classifiers, there are no learnable model parameters (there is only one hyperparameter β which is fixed to 1). Evaluating the evidence $\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m)$ is thus equivalent to computing $\mathbb{P}(\mathbf{Y}_{\mathcal{L}})$ under the BMRF model. This is a combinatorial problem, but we can factorize the joint probability into a chain rule of conditionals and use the same two-stage approximation (TSA) that is employed in [29]. To do so, we consider an arbitrary ordering of the nodes in $\mathcal{L}$ and obtain $p(\mathbf{Y}_{\mathcal{L}})$ with the following chain rule :

$$p(\mathbf{Y}_{\mathcal{L}}) = p(\mathbf{y}_1)p(\mathbf{y}_2|\mathbf{y}_1)...p(\mathbf{y}_{|\mathcal{L}|}|\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_{|\mathcal{L}|-1})\,. \tag{4.12}$$

Each conditional distributions can be evaluated with the approximation provided in [29]. There is no information to determine the label probability of the first node label $p(\mathbf{y}_1)$, so it is set to the uniform distribution. We denote this evidence approximation by

$$\lambda_{TSA,\mathbf{Y}_{\mathcal{L}}} \triangleq \mathbb{P}(\mathbf{Y}_{\mathcal{L}}|TSA). \tag{4.13}$$

.

For the logistic regression model,

$$\mathbb{P}(y_i = k|m, \mathbf{W}) \approx \sigma(\tilde{\mathbf{x}}_i\mathbf{W})^{(k)}. \tag{4.14}$$

For the evidence, $\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m,\mathbf{W}) = \prod_{y_i\in\mathbf{Y}_{\mathcal{L}}} \sigma(\tilde{\mathbf{x}}_i\mathbf{W})^{(k_i)}$, where k_i is the categorical index of y_i. To obtain the evidence, the weight matrix should be integrated out:

$$\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m) = \int_{\mathbf{W}} \prod_{y_i\in\mathbf{Y}_{\mathcal{L}}} \sigma(\tilde{\mathbf{x}}_i\mathbf{W})^{(k_i)}\mathbb{P}(\mathbf{W})\, d\mathbf{W}\,. \tag{4.15}$$

Since this integral is not analytically tractable, $\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m)$ is approximated by the likelihood of the logistic regression model with weights $\mathbf{W}_{\mathbf{Y}_{\mathcal{L}}}$:

$$\mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m) \approx \mathbb{P}(\mathbf{Y}_{\mathcal{L}}|m,\mathbf{W}_{\mathbf{Y}_{\mathcal{L}}}) \tag{4.16}$$

The evidence of the logistic regression model is denoted by:

$$\lambda_{LG,\mathbf{Y}_{\mathcal{L}}} \triangleq \prod_{y_i\in\mathbf{Y}_{\mathcal{L}}} \sigma(\tilde{\mathbf{x}}_i\mathbf{W}_{\mathbf{Y}_{\mathcal{L}}})^{(k_i)}. \tag{4.17}$$

This leads to a sufficiently accurate approximation of the evidence for our purpose (which is just to achieve an adaptive balance between label propagation and graph-cognizant logistic regression). The approximations of the evidences are normalized to sum to one:

$$\bar{\lambda}_{m,\mathbf{Y}_{\mathcal{L}}} = \frac{\lambda_{m,\mathbf{Y}_{\mathcal{L}}}}{\sum_{m'=1}^{M}\lambda_{m',\mathbf{Y}_{\mathcal{L}}}}\,. \tag{4.18}$$

The final query policy is given by the combination of the two risk estimators, balanced by their marginal likelihoods:

$$q^* = \arg\min_{q\in\mathcal{U}} \bar{\lambda}_{LG,\mathbf{Y}_{\mathcal{L}}} R^{+q}_{|\mathbf{Y}_{\mathcal{L}},LG} + \bar{\lambda}_{TSA,\mathbf{Y}_{\mathcal{L}}} R^{+q}_{|\mathbf{Y}_{\mathcal{L}},TSA}\,. \tag{4.19}$$

If the labelled set $\mathcal{L}$ only has one class, then the logistic regression model is not used $\bar{\lambda}_{LG,\mathbf{Y}_{\mathcal{L}}} = 0$ and only the TSA risk is considered for the query selection.

4.4 Experiments

4.4.1 Experiment Settings

To conduct a thorough assessment of how the GEEM algorithm compares to the state-of-the-art methods, three different experiment settings are presented. As in the literature review, the baselines are divided into two groups: the GCN-based models and the BMRF-based models. Although label propagation methods were proposed to address the non-attributed graph setting, they can and should be compared with algorithms that can take advantage of node features.

Since the GCN-based methods involve training a deep learning model, they require more nodes in the initial labelled set. In *Experiment 1*, following the semi-supervised node classification formulation, the nodes are partitioned into train, test, and validation sets. The train set corresponds to the labelled set $\mathcal{L}$, the test and validation sets are withheld from the query algorithm, and the remaining nodes in the graph form the unlabelled set. So the full graph and node features are known, but test and validation nodes cannot be added to the labelled set. During the learning process, the algorithm is aware of the topology of the entire graph (including test and validation nodes); it cannot request labels from the test or validation sets but otherwise it is unaware of which nodes constitute the test set. As mentioned previously, the validation set is an unrealistic assumption in the active learning setting and should technically be included in the labelled set. This impracticality is ignored in the Experiment 1 setting, but the dependence on the validation set of the GCN-based algorithms will be highlighted in the experiment.

Experiment 2 is constructed to resemble the setting that has been traditionally used to evaluate label propagation methods, namely that there is no distinction between the unlabelled set and the test set. *Experiment 3* is designed to highlight the difference between the transductive and inductive settings. Since the label propagation methods are highly dependent on the graph used for training, the label propagation methodologies are not suited to an inductive case at all. In Experiment 3, the test nodes are removed from the graph during the learning process and then reintroduced for testing.

Experiment 1: Initial Labelled Set, Transductive: The three sets $\mathcal{U}, \mathcal{L}, \mathcal{T}$ are obtained by partitioning the nodes $\mathcal{V}$. For the Cora and Citeseer dataset, the nodes in $\mathcal{L}_0$ are chosen at random for each trial and the set is of cardinality equal to 0.5% of the nodes. This is reduced to 0.01% for the larger datasets Amazon-Photo and Amazon-Computers to achieve similar initial set sizes. The test set is formed of 20% of the nodes, again chosen at random for each trial.

Experiment 2: Single Labelled Node, Transductive: The test set coincides with the unlabelled set $\mathcal{U} = \mathcal{T}$ and the initial labelled set only has one node $|\mathcal{L}_0| = 1$, chosen at random for each trial.

Experiment 3: Single Labelled Node, Inductive: The nodes are again partitioned in three sets $\mathcal{U}, \mathcal{L}, \mathcal{T}$, but this time the test nodes are also removed from the graph. The initial labelled set only has one node $|\mathcal{L}_0| = 1$, and the test set is formed from 20% of the nodes. The nodes in these sets are chosen at random and vary from trial to trial.

4.4.2 Datasets

The performance is evaluated on four different benchmark datasets for node classification from [38]. Cora and Citeseer [25] are citation datasets. Nodes represent journal articles and an undirected edge is included when one article cites another. The node features are bag-of-words representations of article content. Amazon-Photo and Amazon-Computers [38] are larger graphs based on customers' co-purchase history records. For each dataset we isolate the largest connected component in the graph following [38]; this also allows us to accommodate the label-propagation baseline. The description of the dataset statistics is shown in Table 4–1.

Table 4–1: Statistics of evaluation datasets.

Dataset	**#Classes**	**#Features**	**#Nodes**	**#Edges**	**Edge density**
Cora	7	1,433	2,485	5,069	0.04%
Citeseer	6	3,703	2,110	3,668	0.04%
Amazon-Comp.	10	767	13,381	245,778	0.07%
Amazon-Photo	8	745	7,487	119,043	0.11%

4.4.3 Baselines and Proposed Algorithms

- **Random**: This baseline chooses a node to query by uniform random selection, and then performs classification using a linear GCN (the same graph-cognizant logistic regression predictive model as is used in our proposed method). This is a standard baseline that can be evaluated in all settings.
- *Experiment 1*:
  - **AGE**: The graph neural network based algorithm proposed in [17].

- **AGE non-optimized**: The graph neural network based algorithm proposed in [17] without fine-tuned hyperparameters. More details are provided in Section 4.4.4.
- **ANRMAB**: The graph neural network algorithm proposed in [6] based on a multi-armed bandit.
- **GEEM***: The proposed algorithm based on graph-cognizant logistic regression and expected error minimization from Section 4.2.

- *Experiments 2 and 3*:
  - **TSA**: The label-propagation algorithm based on a two-stage approximation of the BMRF model [29].
  - **EC-TV, EC-MSD**: The two best performing expected change metrics for the Gaussian random field model proposed in [30].
  - **Combined***: The proposed combined algorithm using Bayesian model averaging to adaptively merge graph-cognizant logistic regression and label propagation in an EEM framework from Section 4.3.

4.4.4 Experimental Details

For each experiment, the average over 20 trials with different random partitions is reported, along with confidence intervals on the means derived using bootstrap. All GCNs and linearized GCNs have 2 layers. The weight-adapting parameter of AGE is set to the values in [17] and to 0.995 for datasets that were not studied in [17]. For Cora and Citeseer, the budget b is set to 60 nodes. For the larger datasets, Am-Photo and Am-Comp, the budget is lowered to 40 and the computational complexity for

GEEM is reduced by evaluating the risk using a subset of 500 nodes, selected randomly in an approach similar to [39]. This has minimal impact on performance. The GCN hyperparameters are set to the values found by [38] to be the best performing hyperparameter configurations for each dataset. Early stopping is not employed because access to a validation set is not a reasonable assumption in an active learning setting. The "non-optimized" version of AGE is included to emulate the case in practice where we would usually not have access to the tuned hyperparameters provided by [38]. For the non-optimized version of AGE, the hyperparameter configuration for each trial was randomly selected from the values considered in the grid search of [38].

4.4.5 Results

To properly assess the performance of an active learning algorithm, the accuracy of the model is reported as nodes are being added to the labelled set. We present figures that show the accuracy as a function of the number of labelled nodes, which can be seen as the size of the training set, from the starting initial labelled set size ($|\mathcal{L}_0|$) until the algorithm reaches the query budget ($|\mathcal{L}_0| + b$). As a result, the different methods are being compared at all points of the process, for the same number of chosen nodes. This is important as it is not only the end performance that matters, but also how quickly the algorithm can increase its accuracy by choosing informative nodes.

Results are presented in this fashion for all three experiments. Figure 4–1 shows the average accuracy, with 5/95 confidence intervals for each baseline corresponding to *Experiment 1* for the four datasets, and Figure 4–2 shows the same experimental

results for the label-propagation baseline TSA for Cora and Citeseer. In Table 4–2, snapshots of accuracies taken during the active learning process of Figures 4–1 are reported, which allows us to present the outcomes of statistical significance tests on the results. The label-propagation baselines for *Experiment 2* and *Experiment 3* are portrayed in Figure 4–3.

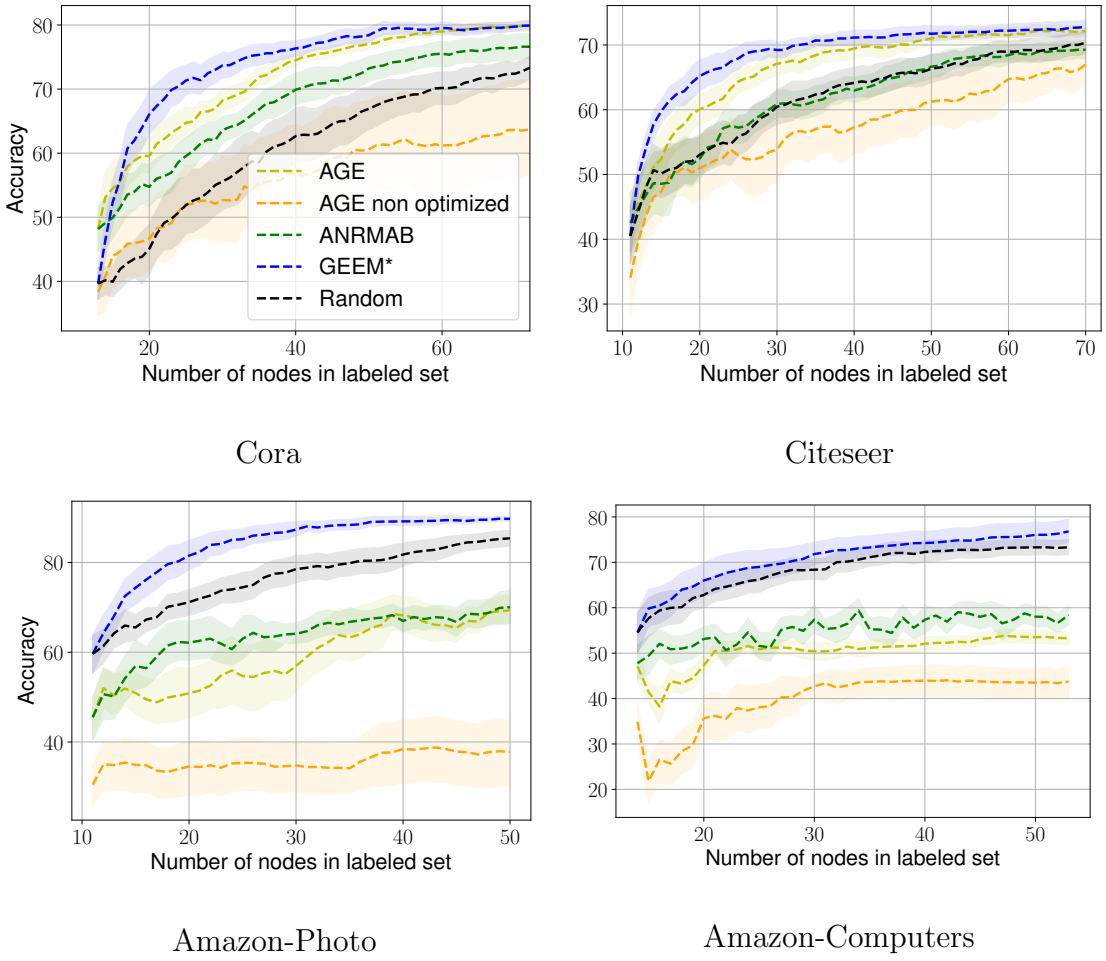
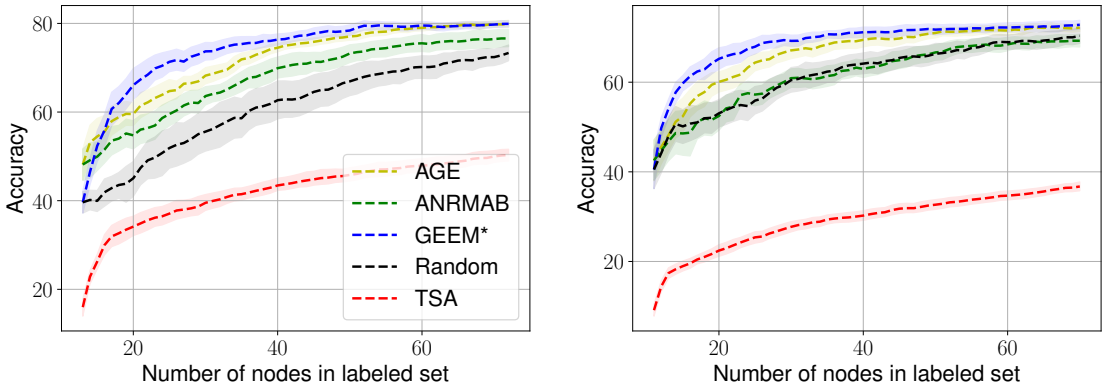


Figure 4–1: Experiment 1. Each point on a curve shows the mean classification accuracy achieved across 20 random partitions after the corresponding algorithm has selected nodes to form an augmented labelled set of size equal to the indicated number of nodes. The shaded regions indicate 5/95 confidence intervals on the means derived using bootstrap.



Cora Citeseer

Figure 4–2: Experiment 1 with the label-propagation baseline TSA. The label propagation method does not transpose well to other experimental settings.

Table 4–2: Experiment 1: Average accuracy at different budgets. Asterisks indicate that a Wilcoxon ranking test showed a significant difference (at the 5% significance level) between the marked method and the best performing baseline.

budget b	**0**	**1**	**10**	**30**	**60**
			Cora		
GEEM*	39.6	46.5	**69.8***	**77.2***	**79.9**
Random	39.6	40.2	49.7	63.0	73.3
AGE	**46.6**	**52.7**	61.6	74.9	79.8
ANRMAB	**46.6**	47.5	59.1	72.7	78.1
			Citeseer		
GEEM*	40.5	**49.7***	**65.8***	**71.2**	**72.8**
Random	40.5	44.1	53.8	64.4	70.4
AGE	**41.2**	44.7	60.5	69.1	71.4
ANRMAB	**41.2**	44.1	55.7	64.6	69.4
budget b	**0**	**1**	**10**	**30**	**40**
			Amazon-Photo		
GEEM*	**59.6**	**64.3**	**82.4***	**89.2***	**90.7***
Random	**59.6**	61.4	72.0	82.3	87.6
AGE	45.5	52.0	51.5	67.8	69.3
ANRMAB	45.5	50.6	62.6	67.8	70.0
			Amazon-Computers		
GEEM*	**54.6**	**59.8**	**68.8***	**74.8***	**76.8***
Random	**54.6**	57.7	65.9	72.8	73.3
AGE	47.1	41.5	51.6	52.4	53.3
ANRMAB	47.1	49.4	54.6	58.7	58.5

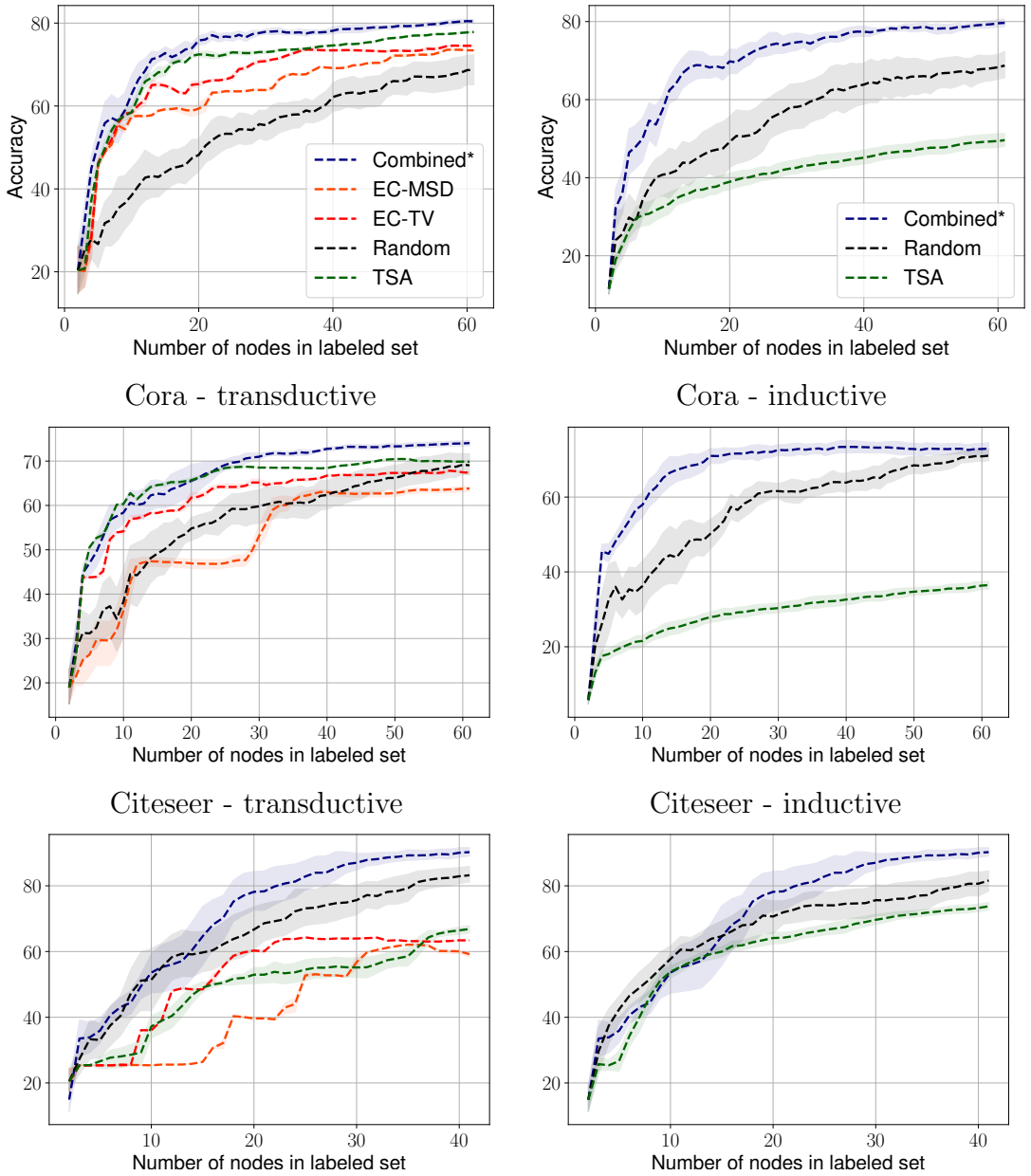


Figure 4–3: Experiment 2 and 3. Performance comparison between the label propagation algorithms and the proposed combined model-averaging expected error minimization method for both the transductive and inductive case.

4.5 Discussion

Experiment 1: For all presented datasets, the proposed algorithms outperform the other GCN-based methods.

At the starting point of the query process ($b = 0$), the query algorithm does not come into play and the more powerful deep learning models are expected to outperform the logistic regression model. This is the case for Cora and Citeseer; as they are very common benchmark datasets for GNNs, their hyperparameters are likely to have been correctly fine-tuned. However for the two other less common datasets, the logistic regression outperforms the GCN by almost 30 percent, even though the GCN hyperparameters are taken from a paper in which hyperparameters are optimized for these datasets [38].

The dependence on the validation set is clearly highlighted by the performance of the non-optimized version of AGE (orange line) that emulates a scenario where no cross-validation can be performed. In all cases, a significant performance deterioration can be observed. The AGE performance curve is mirrored by its non-optimized version, but there is a reduction in accuracy by several percent. In addition, AGE outperforms the ANRMAB algorithm for the datasets where its weight-adapting parameter was tuned (Cora and Citeseer). These results clearly highlight the advantage of using a model that does not rely heavily on a validation set for hyperparameter tuning.

Overall, the GEEM substantially improved accuracy in all cases. For the cases where the logistic regression was initially outperformed by the GCN, once the query process starts the proposed method quickly surpasses the competitors. It even

reaches statistical significance after only 1 query in the case of Citeseer. This shows how effective the query selection of the GEEM is.

In Figure 4–2, the best-performing label propagation technique (TSA [29]) is included for Experiment 1. These results are provided to illustrate that label propagation methods are not competitive with the GNN-based methods when a larger initial training set is available and when a test set of nodes is removed from the candidate query pool.

Experiments 2, 3: Figure 4–3 compares the performance of the proposed Combined method with the label propagation algorithms. In the transductive setting, the proposed method is much better than Random selection. Since it incorporates the TSA technique, its performance is similar to TSA when few nodes have been queried. As the number of labels increases, there starts to be a small but significant improvement in accuracy. The inability of the label propagation methods to adapt to the inductive setting is shown clearly in all cases. In order to choose effective nodes to query, these methods need to know the topology of the entire graph and the graph must include the test nodes. By contrast, the proposed Combined method, which incorporates graph-based logistic regression, achieves similar performance in both inductive and transductive settings.

4.6 Summary

In this chapter, an EEM-based algorithm has been introduced with an extension to cover the extreme case of having only one node as the initial labelled set. The efficacy has been demonstrated through extensive experiments. A thorough assessment of how the proposed approach compares to state-of-the-art baselines has been made.

The results underline the weaknesses of the two identified types of baseline and show the dominance of the proposed GEEM method, which can be advantageous in all settings. A notable drawback is the run-time of the methods that are based on the EEM framework. For the Cora dataset, for example, the time required to identify one query of the GEEM algorithm is approximately seven times that of the GCN-based algorithms, and four to five times that of the label-propagation baselines. In the following chapter, this limitation is indirectly addressed by looking at the problem from a different perspective. This leads to a modified version of the algorithm that can profit from this new problem formulation.

CHAPTER 5
Preemptive GEEM

This chapter presents a variation of the GEEM algorithm. A new problem statement prompted by a closer examination of a practical active learning process timeline is introduced in the first section. This formulation leads to the proposal of an approximate algorithm designed to be more time efficient. Experiments, analysis and discussion are jointly presented in Section 5.4, followed by the results that show that the performance loss caused by the approximation is marginal. Finally, theoretical bounds on the approximation are presented in the last section of the chapter.

5.1 A Modified Problem Setting

5.1.1 Motivation

The query step of active learning involves the following steps: 1) select the point to query; and 2) add its label to the labelled set. In most numerical experiments that emulate an active learning scenario, no oracle is actually being queried for the label. Usually the complete set of labels is readily available, which makes the transition between the two steps instantaneous. This obfuscates an important practical reality that is at the core of the motivation of active learning; in practice, label acquisition is tedious and requires considerable resources. This often translates into labelling being time consuming for a human oracle. It implies that the active learning algorithm stalls between 1) and 2), waiting for the oracle to label q_t^*. In addition, the oracle must wait while the algorithm computes the best subsequent query node q_{t+1}^*. This

is inefficient for both parties and, for a human oracle, frustrating. In [40], this exact issue is discussed:

> Since the query has been labeled and added to the training set $\mathcal{L}$, the learner re-trains using this newly-acquired knowledge and selects another single query, given all the previously labeled instances, and the process repeats. However, this is not always a realistic setting. For many applications, the process of inducing a model from training data may be slow or expensive, which is often the case with state-of-the-art methods like large ensemble algorithms, or the graphical models used for structured-prediction tasks. In such cases, it is an inefficient use of labeling resources (e.g., a human annotator's time) to wait for the model to re-train before querying for the next label. [40, p. 3]

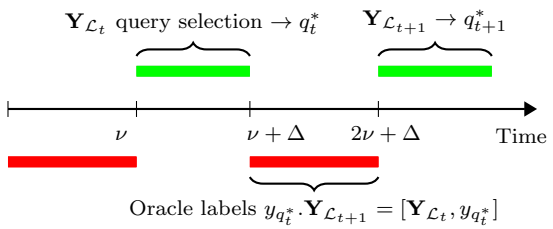
Settles used this observation in [40] as a motivation for batch querying, where the algorithm selects multiple instances instead of only one to make the process more efficient. However this only reduces the number of times that the oracle is stalled; the root problem remains. The approach taken in this thesis is different. Our first step is to explicitly identify the labeling time component in the problem formulation.

5.1.2 Formulation

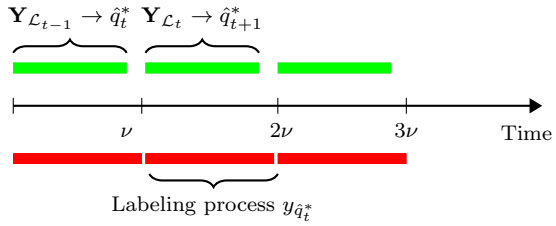
1. **Prediction Step** : Form a prediction of $\hat{\mathbf{Y}}_{\mathcal{T}_t}$ based on the graph, $\mathcal{G}$, features, $\mathbf{X}$, and the current labels, $\mathbf{Y}_{\mathcal{L}_t}$;
2. **Query Step** : Until the budget is exhausted, select a node $q_t^* \in \mathcal{U}_t$ to query and to add to the labelled set.

3. **Labeling Step** : The oracle takes time Δ to label q_t^*. We update the sets: $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{q_t^*\}$, $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{q_t^*\}$.

Our main idea is then to derive an active learning algorithm that can identify the query node q_t^* using the label set $\mathcal{L}_{t-1}$. If the labelling time and the query computation time are similar, then neither the oracle nor the algorithm stalls for long. While the oracle is generating the label for q_t^*, this *preemptive* active learning algorithm identifies in parallel the best query node q_{t+1}^* using $\mathcal{L}_t$. Figure 5–1 compares the timelines of the standard single-query active learning procedure (in which the query generation algorithm waits for the oracle and vice versa) with the preemptive strategy where labelling and query generation are performed in parallel.



(a) Timeline for the standard active learning process.



(b) Timeline for preemptive active learning process.

Figure 5–1: A comparison of the timelines of the standard single-query active learning process and the proposed preemptive process.

5.2 Preemptive Query (PreGEEM)

In this section, the GEEM algorithm is adapted to perform preemptive query calculation, using the labelling time to identify the next node to query. Instead of waiting for the oracle to label q^*_{t-1} to start the identification of q^*_t during iteration t, the algorithm forms an approximation of the risk before knowing $y_{q^*_{t-1}}$. The direct approach is to replace the risk $R^{+q_t}_{|\mathbf{Y}_{\mathcal{L}_t}}$ with the expectation over the possible values of $y_{q^*_{t-1}}$, but this increases the computational complexity by a factor of $|\mathcal{K}|$, which is highly undesirable. To avoid this penalty, $R^{+q}_{|\mathbf{Y}_{\mathcal{L}_{t-1}},q^*_{t-1}}$ is further approximated using the value of risk for the label at the mode of $\mathbb{P}(y^*_{q_{t-1}}|\mathbf{Y}_{\mathcal{L}_{t-1}})$. Effectively, the predicted label $\hat{y}_{q^*_{t-1}}$ of the previous model $\mathbb{P}(\cdot|\mathbf{Y}_{\mathcal{L}_{t-1}})$ is added to the labelled set to form an augmented set $\mathbf{Y}'_{\mathcal{L}_t} = \{\mathbf{Y}_{\mathcal{L}_{t-1}} \cup \{\hat{y}_{q^*_{t-1}}\}\}$ and define an approximate risk:

$$\hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} \triangleq E_{y_q}\left[E_{\mathbf{Y}_{\mathcal{U}_t^{-q}}}\left[\tfrac{1}{|\mathcal{U}_t^{-q}|}\sum_{i\in\mathcal{U}_t^{-q}} \mathbb{1}[\hat{y}_i \neq y_i|y_q, \mathbf{Y}'_{\mathcal{L}_t}]\right]\right]. \tag{5.1}$$

As was the case for GEEM in equation 4.9, the probability $\mathbb{P}(y = k|\mathbf{Y}'_{\mathcal{L}_t})$ is approximated by the model prediction $\sigma(\tilde{\mathbf{x}}_q\mathbf{W}_{\mathbf{Y}'_{\mathcal{L}_t}})^{(k)}$, however now the $\mathbf{Y}'_{\mathcal{L}_t}$ subscript of the weights indicates that the model is trained on the labelled set containing a predicted label. The same applies for $\varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}} = (1 - \max_{k'\in\mathcal{K}} \sigma(\tilde{\mathbf{x}}_i\mathbf{W}_{\mathbf{Y}'_{\mathcal{L}_t},+q,y_k})^{(k')})$. The approximated risk is then evaluated as :

$$\hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} = \tfrac{1}{|\mathcal{U}_t^{-q}|}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}\sigma(\tilde{\mathbf{x}}_q\mathbf{W}_{\mathbf{Y}'_{\mathcal{L}_t}})^{(k)}. \tag{5.2}$$

The query node is then $\hat{q}^*_t = \arg\min_{q\in\mathcal{U}} \hat{R}^{+q}_{|\mathbf{Y}_{\mathcal{L}'_t}}$.

5.3 Experiment and Results

Since the PreGEEM is an approximation, the main question is how much this impacts the performance of the algorithm compared to GEEM. To answer this, the same *Experiment 1* from Section 4.4 is conducted for all datasets and the PreGEEM performance is compared to the GEEM. The results are presented in Figures 5–2 and Table 5–1.

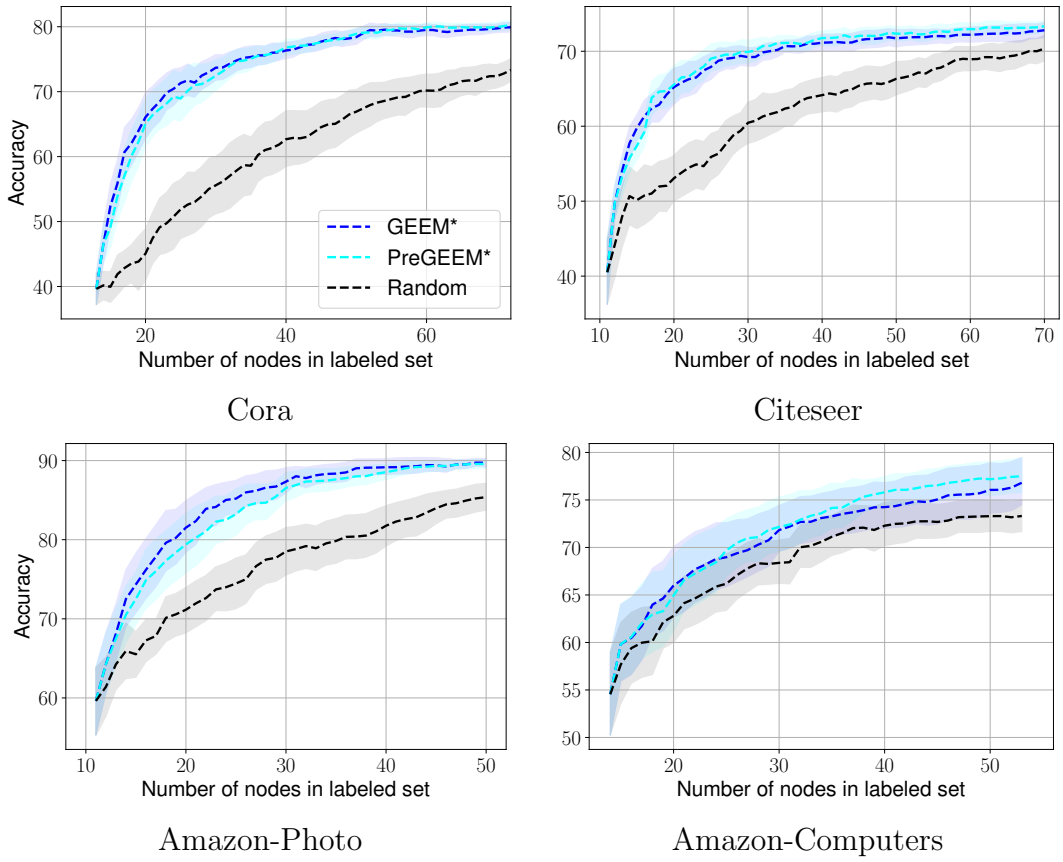


Figure 5–2: Experiment 1 for a GEEM vs PreGEEM comparison. Each point on a curve shows the mean classification accuracy achieved across 20 random partitions after the corresponding algorithm has selected nodes to form an augmented labelled set of size equal to the indicated number of nodes. The shaded regions indicate 5/95 confidence intervals on the means derived using bootstrap.

Table 5–1: Experiment 1: Average accuracy at different budgets, comparing GEEM to PreGEEM. Asterisks indicate that a Wilcoxon ranking test showed a significant difference (at the 5% significance level) between GEEM and PreGEEM.

budget b	**0**	**1**	**10**	**30**	**60**
			Cora		
GEEM	39.6	**46.5**	**69.8**	**77.2**	79.9
PreGEEM	39.6	**46.5**	68.2	77.1	**80.3**
Random	39.6	40.2	49.7	63.0	73.3
			Citeseer		
GEEM	40.5	**49.7**	65.8	71.2	72.8
PreGEEM	40.5	**49.7**	**66.5**	**71.8**	**73.3**
Random	40.5	44.1	53.8	64.4	70.4
budget b	**0**	**1**	**10**	**30**	**40**
			Amazon-Photo		
GEEM	59.6	**64.3**	**82.4**	**89.2**	**90.7***
PreGEEM	59.6	**64.3**	80.3	88.8	89.6
Random	59.6	61.4	72.0	82.3	87.6
			Amazon-Computers		
GEEM	54.6	**59.8**	**68.8**	74.8	76.8
PreGEEM	54.6	**59.8**	68.4	**76.5**	**77.5**
Random	54.6	57.7	65.9	72.8	73.3

5.4 Discussion

By inspecting the results in Figure 5–2, it is clear that the approximation has little effect. In most cases, the performance of GEEM is marginally better, but not always, and the difference is small. Table 5–1 confirms this finding by identifying no statistically significant difference between the performance of the two algorithms at any point except for one.

To provide further insight into the effect of the approximation, I now present a case study that follows the risks from both algorithms as query nodes are added.

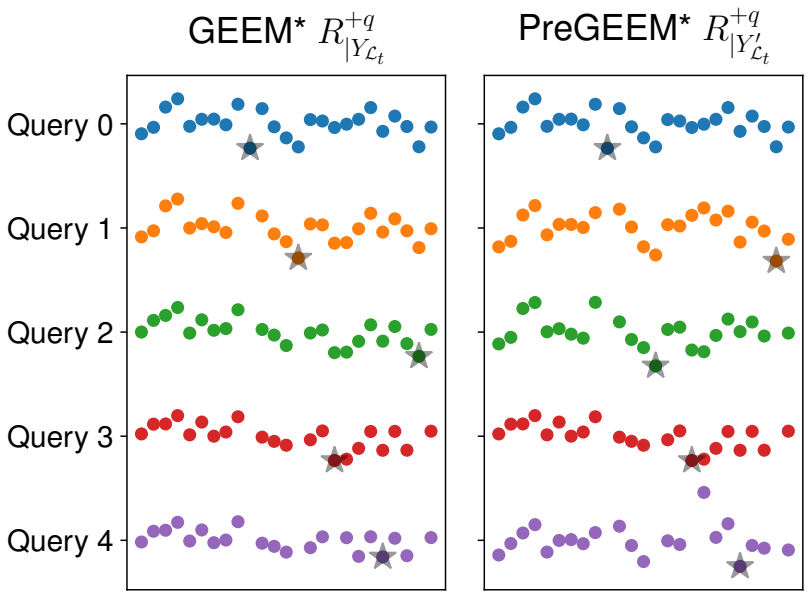


Figure 5–3: Risk comparison for GEEM vs PreGEEM. This diagram follows the risk computations for 25 nodes in the Cora dataset for one trial. The black star indicates which node was selected (following the algorithm, it is the one with the lowest expected risk).

For PreGEEM to yield poorer results compared to GEEM, three things need to happen. First, $\mathbf{Y}_{\mathcal{L}_t}$ needs to be different from $\mathbf{Y}'_{\mathcal{L}_t}$. This can be caused by the previous query being different, thus changing $\mathcal{L}_{t-1}$, or an error in the prediction of $\hat{y}_{q^*_{t-1}}$. If neither of these occur, there is no difference between GEEM and PreGEEM calculations and the risks would be identical. The second element is that the difference in risk values must be substantial enough to change the ordering of the $\hat{R}^{+q}_{|\mathbf{Y}_{\mathcal{L}'_t}}$; this will lead to a different query $\hat{q}^*_t \neq q^*_t$. It is possible that, even if the risk values are not matching, the same node minimizes both sets of risk values. Lastly, this

resulting changed labelled set $\mathcal{L}$ should actually be poorer for model generalization, which manifests itself in lower test accuracy.

Figure 5–3 displays the evolution of the two active learning algorithm risk values for a small subset of nodes. When identifying the first query, the labels of the initial set $\mathcal{L}_0$ are accessible so no approximation is being made and the same node is selected by both algorithms. This is reflected by the GEEM and PreGEEM having perfectly matching blue point patterns. Subsequently, the inference algorithm makes a mistake in $\hat{y}_{q_0^*}$ for PreGEEM, which cause the risk values to differ. The impact is small, but enough to change the node that minimizes $\hat{R}^{+q}_{|\mathbf{Y}_{\mathcal{L}_1'}}$ and PreGEEM selects a different query. This deviation is corrected at the third step, where the two algorithms switch their query selection. Both labelled sets $\mathcal{L}_3$ contain the same nodes. During the selection of the fourth query node, no mistake is made by the inference algorithm for PreGEEM and the risk values are identical. For the last query, a mistake is made but this time it is not enough to sway the selection process to a different node. After five steps the same nodes have been selected by both algorithms. This demonstrates how the approximation can have a minor impact on the classification performance.

5.5 Bounds on the PreGEEM Risk Error

Empirical analysis from the previous section clearly showed that there is no significant difference in the performance of GEEM vs PreGEEM. A key observation made in the case study was that a deviation caused by the PreGEEM approximation needed to be important enough in order to modify the query and potentially negatively impact the performance. This motivates our interest in bounding the magnitude of the error on the risk of PreGEEM.

As explained in Section 5.2, PreGEEM relies on a prediction rather than waiting for the true label to compute the risk. So an error only arises when there is a misclassification. The bounds presented in this Section focus on the error incurred by labelled sets that differ by one label: $|\hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} - R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}}|$. The formulations for the different risks are given in equation 4.9 for GEEM, and in equation 5.2 for PreGEEM. The difference or error arises because the prediction values are made by logistic regression models that are trained on slightly different training sets:

$$|\hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} - R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}}| \leq \left| \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}} \sigma(\tilde{\mathbf{x}}_q \mathbf{W}_{\mathbf{Y}'_{\mathcal{L}_t}})^{(k)} - \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \sigma(\tilde{\mathbf{x}}_q \mathbf{W}_{\mathbf{Y}_{\mathcal{L}_t}})^{(k)} \right| \quad (5.3)$$

As a result, the bound is constructed as follows: 1) bound the difference between the regression weights learned from different label sets; 2) bound the difference in the predictions made on the same point by two logistic regression models; and 3) bound the elements in the sum of the risk. 1) and 2) can be viewed as preliminary steps and are addressed first in Section 5.5.1. Then, the bound 3) is developed for two cases. For clarity, the binary classification task is first considered in section 5.5.2; then a more general bound for multiclass classification is stated in Section 5.5.3. Section 5.5.4 presents two peripheral lemmas that are used in the proofs of the main results.

5.5.1 Preliminaries

The first step is to bound the difference in the logistic regression weights derived for the two label sets. In [41], Sivan et al. consider the problem of online learning, where the available data is being gradually replaced and updated. In their context, the two datasets are derived by applying a sliding window to the same stream of data. Sivan et al. are interested in how much the weights trained on an updated dataset can differ from the weights learned using the previous dataset, given that the datasets contain many common examples. A bound on this difference can allow one to make a more informed decision about whether to retrain or not. There are analogies between this problem formulation and our setting and the main theorem in [41] provides the starting point for the development of our bounds.

We start by stating the result of [41]. Let $\{\mathbf{x}_i \in \mathbb{R}^{1\times d} : 1 \leq i \leq N\}$ be the feature vectors of N instances, indexed by $\mathcal{D}$. Sivan et al. define the optimization problem P of learning the weights $\mathbf{w}_p$ of a logistic regression model by minimizing the sum of the loss of individual instances $\ell_i(\mathbf{w})$ under L^2 regularization with associated regularization parameter C_P.

$$P : \mathbf{w}_p = \underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min}\, C_p \sum_{i\in\mathcal{D}} \ell_i(\mathbf{w}) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.4}$$

Let $\mathbf{w}_1$ be the optimal solution of P_1 over the labelled samples with indices $\mathcal{D}_1$, and let $\mathbf{w}_2$ be the solution of P_2 over the labelled samples $\mathcal{D}_2$. Similarly, C_1 is associated to P_1 and C_2 to P_2. In [41], Sivan et al. process a dataset that is being updated, so they define a set of indices of added samples, $\mathcal{D}_\mathcal{A} = \mathcal{D}_2 \backslash \mathcal{D}_1$, and a set of indices of

samples removed, $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_1 \backslash \mathcal{D}_2$. Finally, let Δg be :

$$\Delta g := \sum_{i \in \mathcal{D}_{\mathcal{A}}} \nabla \ell_i \left(\mathbf{w}_1 \right) - \sum_{i \in \mathcal{D}_{\mathcal{R}}} \nabla \ell_i \left(\mathbf{w}_1 \right) \tag{5.5}$$

Theorem 1 of [41] establishes that the distance between $\mathbf{w}_1$ and $\mathbf{w}_2$ is bounded by: $\|\mathbf{w}_1 - \mathbf{w}_1\| \leq 2\|\mathbf{r}\|$, where

$$\mathbf{r} = \frac{1}{2} \left(\mathbf{w}_1 - \frac{C_2}{C_1} \mathbf{w}_1 + C_2 \Delta g \right)$$

We can see that this result can be directly applied to our setting. Rather than an updated dataset, we have two datasets that share most of the elements, with some instances that differ by being associated with different binary class labels $y_{i,1}$ and $y_{i,2}$. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two datasets that contain all N instances, with label values that differ for exactly $M < N$ of the feature vectors, with indices $\{j_m\}, m = 1, \ldots, M$. By fixing the regularization parameter to the same λ for both models and defining the loss function as being the cross entropy, $\ell(\mathbf{x}_i, y_i; \mathbf{w}) = -\Big(y_i \log \sigma(\mathbf{x}_i \mathbf{w}) + \big(1 - y_i\big) \log \big(1 - \sigma(\mathbf{x}_i \mathbf{w})\big) \Big)$, where $\sigma(x) = \dfrac{1}{1 + e^{-x}}$ is the sigmoid function, we have:

$$\begin{aligned}
\mathbf{r} &\triangleq \frac{1}{2\lambda} \sum_{m=1}^{M} \left(\nabla_{\mathbf{w}} \ell(\mathbf{x}_{j_m}, y_{j_m,1}; \mathbf{w})|_{\mathbf{w}=\mathbf{w}_1} - \nabla_{\mathbf{w}} \ell(\mathbf{x}_{j_m}, y_{j_m,2}; \mathbf{w})|_{\mathbf{w}=\mathbf{w}_1} \right), \\
&= \frac{1}{2\lambda} \sum_{m=1}^{M} \left(- \big(y_{j_m,1} - \sigma(\mathbf{x}_{j_m} \mathbf{w}_1)\big) \mathbf{x}_{j_m} + \big(y_{j_m,2} - \sigma(\mathbf{x}_{j_m} \mathbf{w}_1)\big) \mathbf{x}_{j_m} \right), \\
&= \frac{1}{2\lambda} \sum_{m=1}^{M} \big(y_{i_k,2} - y_{i_k,1}\big) \mathbf{x}_{i_k} \,.
\end{aligned} \tag{5.6}$$

We can then use $||\mathbf{w}_1 - \mathbf{w}_2|| \leq 2||\mathbf{r}||$ in our setting.

Lemma 1 from [41] provides upper and lower bounds for $\mathbf{x}_j\mathbf{w}_2$ in terms of the other weights $\mathbf{w}_1$. We have:

$$\mathbf{x}_j\mathbf{w}_1 - \mathbf{x}_j\mathbf{r} - ||\mathbf{r}||.||\mathbf{x}_j|| \leq \mathbf{x}_j\mathbf{w}_2 \leq \mathbf{x}_j\mathbf{w}_1 - \mathbf{x}_j\mathbf{r} + ||\mathbf{r}||.||\mathbf{x}_j|| \,. \tag{5.7}$$

5.5.2 Bound on Risk Error: Binary Classification

Based on the results outline above, the following proposition can be specified, bounding the difference between the predicted values of the two models.

Proposition 1. *Let* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *be the weights derived by* L^2*-penalized logistic regression for two datasets* $(\mathbf{X}, \mathbf{Y})$, $(\mathbf{X}, \mathbf{Y}')$*, with common feature vectors but label sets differing for* M *vectors indexed by* $\{j_m\}, m = 1, \ldots, M$*. Define* $\eta \triangleq \frac{1}{2\lambda}\sum_{m=1}^{M}||\mathbf{x}_{j_m}||$ *and* $b_{\pm\eta}(\mathbf{w}_2, i) \triangleq \sigma(\mathbf{x}_i\mathbf{w}_2) - \sigma(\mathbf{x}_i\mathbf{w}_2 \pm 2\eta||\mathbf{x}_i||)$*. Then for any* $\mathbf{x}_i$*, for* $i \in \{1, \ldots, N\}$*:*

$$|\sigma(\mathbf{x}_i\mathbf{w}_1) - \sigma(\mathbf{x}_i\mathbf{w}_2)| \leq \max(|b_{+\eta}(\mathbf{w}_2, i)|, |b_{-\eta}(\mathbf{w}_2, i)|) \,. \tag{5.8}$$

Proof. We first note that $\left(y_{i_k,2} - y_{i_k,1}\right) \in \{-1, 1\}$ implies from the definition of $\mathbf{r}$ in (5.6) that $||\mathbf{r}|| \leq \dfrac{1}{2\lambda}\displaystyle\sum_{m=1}^{M}||\mathbf{x}_{j_m}|| = \eta$. Applying the Cauchy Schwarz inequality to (5.7), we have:

$$\mathbf{x}_i\mathbf{w}_2 - 2||\mathbf{r}||.||\mathbf{x}_i|| \leq \mathbf{x}_i\mathbf{w}_1 \leq \mathbf{x}_i\mathbf{w}_2 + 2||\mathbf{r}||.||\mathbf{x}_i|| \,, \tag{5.9}$$

and hence

$$\mathbf{x}_i\mathbf{w}_2 - 2\eta||\mathbf{x}_i|| \leq \mathbf{x}_i\mathbf{w}_1 \leq \mathbf{x}_i\mathbf{w}_2 + 2\eta||\mathbf{x}_i|| \,. \tag{5.10}$$

Since $\sigma(\cdot)$ is monotonically increasing, we have $\sigma(\mathbf{x}_i\mathbf{w}_1 - 2\eta||\mathbf{x}_i||) \leq \sigma(\mathbf{x}_i\mathbf{w}_2) \leq \sigma(\mathbf{x}_i\mathbf{w}_1 + 2\eta||\mathbf{x}_i||)$ and (5.8) follows. □

The following result characterizes the potential risk error when we perform L^2-regularized binary logistic regression on $\{\mathbf{Y}'_{\mathcal{L}_t} \cup \{y_q = k\}\}$ to derive weights $\mathbf{w}_{2,k}$ for $k \in \{0, 1\}$. We are interested in bounding the difference in the risks that can arise when we use two label sets $\mathbf{Y}_{\mathcal{L}_t}$ and $\mathbf{Y}'_{\mathcal{L}_t}$ that differ by only one label, associated with the node index q^*_{t-1}. The previous result can directly be applied by associating $\mathcal{D}_1$ with the dataset that contains all true labels $\mathbf{Y}_{\mathcal{L}_t}$ and $\mathcal{D}_2$ with the available dataset $\mathbf{Y}'_{\mathcal{L}_t}$ that contains the predicted label.

We start by defining the two quantities:

$$\begin{aligned} \eta_q &\triangleq \tfrac{1}{2\lambda}\left(||\tilde{\mathbf{x}}_q|| + ||\tilde{\mathbf{x}}_{q^*_{t-1}}||\right), \\ \tilde{b}(\eta_q, \mathbf{w}_2, i) &\triangleq \max_{k\in\{0,1\}} \max\left(|b_{+\eta_q}(\mathbf{w}_{2,k}, i)|, |b_{-\eta_q}(\mathbf{w}_{2,k}, i)|\right). \end{aligned} \tag{5.11}$$

Recall the definition of the risk of querying node q given a current label set $\mathbf{Y}_{\mathcal{L}_t}$, for the specific case of binary classification:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} \triangleq \tfrac{1}{|\mathcal{U}_t^{-q}|} \sum_{k\in\{0,1\}} \sum_{i\in\mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \mathbb{P}(y_q = k|\mathbf{Y}_{\mathcal{L}_t}), \tag{5.12}$$

where

$$\begin{aligned} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} &\triangleq \Big(1 - \max_{k'\in\{0,1\}} \mathbb{P}(y_i = k'|\mathbf{Y}_{\mathcal{L}_t}, y_q = k)\Big) \\ &= \min_{k'\in\{0,1\}} \mathbb{P}(y_i = k'|\mathbf{Y}_{\mathcal{L}_t}, y_q = k). \end{aligned} \tag{5.13}$$

We now state the main result:

Theorem 5.5.1. *The risk error arising from applying binary L^2-regularized logistic regression with regularization parameter λ to two labelled datasets $(\mathbf{X}, \mathbf{Y}_{\mathcal{L}_t})$ and $(\mathbf{X}, \mathbf{Y}'_{\mathcal{L}_t})$ that differ by one label, associated with the node q^*_{t-1}, is bounded as:*

$$|R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - \hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}}| \leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \tilde{b}(\eta_q, \mathbf{w}_2, i)\,.$$

Proof. We define the random variable $\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}$ which takes value $\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}}$ with probability $\mathbb{P}(y_q = k|\mathbf{Y}_{\mathcal{L}_t})$ for $k \in \{0,1\}$. Analogously, let $\varphi^{+q}_{i,\mathbf{Y}'_{\mathcal{L}_t}}$ take value $\varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}$ with probability $\mathbb{P}(q = k|\mathbf{Y}'_{\mathcal{L}_t})$ for $k \in \{0,1\}$. The difference in risk is then:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - \hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} = \tfrac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}] - E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}'_{\mathcal{L}_t}}]\,, \tag{5.14}$$

where E_{y_q} denotes expectation over y_q conditioned on the corresponding observed label sets, either $\mathbf{Y}_{\mathcal{L}_t}$ or $\mathbf{Y}'_{\mathcal{L}_t}$. For query node q, for each $k_1, k_2 \in \{0,1\}$, we learn weights $\mathbf{w}_{1,k_1}$ using $\{\mathbf{Y}_{\mathcal{L}_t} \cup \{y_q = k_1\}\}$ and weights $\mathbf{w}_{2,k_2}$ using $\{\mathbf{Y}'_{\mathcal{L}_t} \cup \{y_q = k_2\}\}$. For each $i \in \mathcal{U}_t^{-q}$, we have:

$$\begin{aligned}
&|\varphi^{+q}_{i,k_1,\mathbf{Y}_{\mathcal{L}_t}} - \varphi^{+q}_{i,k_2,\mathbf{Y}'_{\mathcal{L}_t}}| \leq |\sigma(\mathbf{x}_i\mathbf{w}_{1,k_1}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k_2})|\,, \text{ using Lemma (5.5.4)}\,,\\
&\leq \max(|\sigma(\mathbf{x}_i\mathbf{w}_{2,k_2}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k_2} - 2\eta_q||\mathbf{x}_i||)|, |\sigma(\mathbf{x}_i\mathbf{w}_{2,k_2}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k_2} + 2\eta_q||\mathbf{x}_i||)|)\,,\\
&\leq \max_{k\in\{0,1\}} \max(|\sigma(\mathbf{x}_i\mathbf{w}_{2,k}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k} - 2\eta_q||\mathbf{x}_i||)|, |\sigma(\mathbf{x}_i\mathbf{w}_{2,k}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k} + 2\eta_q||\mathbf{x}_i||)|)\,.
\end{aligned} \tag{5.15}$$

Here the second inequality follows from Proposition 1, observing that the labels can differ for nodes q^*_{t-1} and q.

Now, the difference in risk is bounded as

$$
\begin{aligned}
|R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - R^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}}| &\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} |E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}] - E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}'_{\mathcal{L}_t}}]|\,, \\
&\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \max(|\max_{k\in\{0,1\}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} - \min_{k\in\{0,1\}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}|, \\
&\qquad |\min_{k\in\{0,1\}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} - \max_{k\in\{0,1\}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}|)\,, \text{ using Lemma (5.5.3)}\,, \\
&\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \max_{k\in\{0,1\}} \max(|\sigma(\mathbf{x}_i\mathbf{w}_{2,k}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k} - 2\eta_q||\mathbf{x}_i||)|, \\
&\qquad |\sigma(\mathbf{x}_i\mathbf{w}_{2,k}) - \sigma(\mathbf{x}_i\mathbf{w}_{2,k} + 2\eta_q||\mathbf{x}_i||)|)\,, \text{ using (5.15)}\,, \\
&= \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \tilde{b}(\eta_q, \mathbf{w}_2, i) \qquad (5.16)
\end{aligned}
$$

□

5.5.3 Bound on Risk Error: Multiclass Classification

Now that the binary case has been presented, we would like to derive a similar result for the more general case of multiclass classification. Every dataset used to evaluate the performance of the proposed algorithms has three classes or more. Extending the theoretical result is not straightforward since a central part of the proof relies on the mathematical formulation of the binary logistic regression prediction model, which does not apply to the multiclass case. In practice, multiclass classification is often solved using a one-versus-all approach, i.e., by repeatedly applying binary logistic regression. This is the method adopted by the solvers we have used to generate the numerical results presented in the thesis. Effectively, the algorithm

performs binary logistic regression for each class (versus all other classes) and then normalizes the sum of the sigmoid outputs to obtain the final prediction. In deriving a bound for multiclass classification, we assume that the one-vs-all method is employed.

As before, we focus on the case where two labels can be different for the proposed query node q and the previous query node q^*_{t-1}. For a given label set $\mathbf{Y}_{\mathcal{L}_t}$, we learn weights $\mathbf{w}_{(k)}$ for each class $k \in \mathcal{K}$ using L^2-regularized binary one-vs-all logistic regression. The output prior to normalization for a given feature vector $\mathbf{x}_i$ is given by $\sigma(\mathbf{x}_i\mathbf{w}_{(k)})$. We then normalize by dividing by $C_i = \sum_{k\in\mathcal{K}} \sigma(\mathbf{x}_i\mathbf{w}_{(k)})$ to obtain a probability vector.

Again we start by defining several quantities. Let $\rho(\eta, \mathbf{W}_2, i)$ be the multiclass version of $\tilde{b}(\eta_q, \mathbf{W}_2, i)$ defined in Equation 5.11. $b_{\pm\eta}(\mathbf{W}_2, i)$ remains the same as defined in Proposition 1. We have:

$$\rho(\eta, \mathbf{W}_2, i) \triangleq \max_{k\in\mathcal{K}} \max(|b_{+\eta}(\mathbf{w}_{2,(k)}, i|, |b_{-\eta}(\mathbf{w}_{2,(k)}, i)|) \tag{5.17}$$

$$\begin{aligned}\beta(\mathbf{W}_2, k, \eta, i) \triangleq \max\Big(&\left|\frac{\sigma(\mathbf{x}_i\mathbf{w}_{2,(k)})}{C_{2,i}} - \frac{\sigma(\mathbf{x}_i\mathbf{w}_{2,(k)}) - \rho(\eta, \mathbf{W}_2, i)}{C_{2,i} + 4\rho(\eta, \mathbf{W}_2, i)}\right|, \\ &\left|\frac{\sigma(\mathbf{x}_i\mathbf{w}_{2,(k)})}{C_{2,i}} - \frac{\sigma(\mathbf{x}_i\mathbf{w}_{2,(k)}) + \rho(\eta, \mathbf{W}_2, i)}{C_{2,i} - 4\rho(\eta, \mathbf{W}_2, i)}\right|\Big),\end{aligned} \tag{5.18}$$

$$\beta(\mathbf{W}_2, \eta, i) \triangleq \max_{k\in\mathcal{K}} \beta(\mathbf{W}_2, k, \eta, i), \tag{5.19}$$

The following proposition is similar to Proposition 1, but applies to the multiclassification case and fixes the number of differing labels to two: $M = 2$. This leads to $\eta = \frac{1}{2\lambda}(||\mathbf{x}_{j_1}|| + ||\mathbf{x}_{j_2}||)$.

Proposition 2. *Let* $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *be the weights derived by multiclass logistic regression for classes* $k \in \mathcal{K}$ *to two datasets* $(\mathbf{X}, \mathbf{Y})$, $(\mathbf{X}, \mathbf{Y}')$ *with common feature vectors but label sets differing for two vectors,* $\mathbf{y}_{i_1} \neq \mathbf{y}'_{i_1}$ *and* $\mathbf{y}_{i_2} \neq \mathbf{y}'_{i_2}$. *Assume that the multiclass logistic regression is performed by conducting* L^2*-regularized one-vs-all binary logistic regression for each class and then normalizing the output values to sum to one. Let* $\mathbf{w}_{1,(k)}$ *denote the weights learned for class* k *using label set* $\mathbf{Y}$ *and* $\mathbf{w}_{2,(k)}$ *denote the corresponding weights learned using label set* $\mathbf{Y}'$. *Let* $p_{1,k,i} = \sigma(\mathbf{x}_i \mathbf{w}_{1,(k)})/C_{1,i}$ *be the output probability associated with class* k *for feature vector* $\mathbf{x}_i$ *using label set* $\mathbf{Y}$, *where* $C_{1,i} = \sum_{k \in \mathcal{K}} \sigma(\mathbf{x}_i \mathbf{w}_{1,(k)})$. *Let* $p_{2,k,i}$ *and* $C_{2,i}$ *be the corresponding entities derived using label set* $\mathbf{Y}'$. *For any feature vector* $\mathbf{x}_i$, *for each class* k*:*

$$|p_{1,k,i} - p_{2,k,i}| \leq \beta(\mathbf{W}_2, \eta, i) \tag{5.20}$$

Proof. Two different labels can affect the weight vectors $\mathbf{w}_{(k)}$ for at most four classes, because there is no change in the one-versus-all binary labels for class k unless at least one of $\mathbf{y}_{j_1}$, $\mathbf{y}_{j_2}$, $\mathbf{y}'_{j_1}$ or $\mathbf{y}'_{j_2}$ equals k. For any class k, at most two labels can change for the binary classification. For the one-vs-all binary classifier for class k, we learn a weight vector $\mathbf{w}_{1,(k)}$ using $\mathbf{Y}$ and $\mathbf{w}_{2,(k)}$ using $\mathbf{Y}'$. Then from Proposition 1, we directly have (for the case where both labels change):

$$|\sigma(\mathbf{x}_i \mathbf{w}_{1,(k)}) - \sigma(\mathbf{x}_i \mathbf{w}_{2,(k)})| \leq \max(|b_{+\eta}(\mathbf{w}_{2,(k)}, i)|, |b_{-\eta}(\mathbf{w}_{2,(k)}, i)|)\,. \tag{5.21}$$

Using $\rho(\eta, \mathbf{W}_2, i)$ defined in Equation (5.17), we have for any class k:

$$\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)}) - \rho(\eta, \mathbf{W}_2, i) \leq \sigma(\mathbf{x}_i \mathbf{w}_{1,(k)}) \leq \sigma(\mathbf{x}_i \mathbf{w}_{2,(k)}) + \rho(\eta, \mathbf{W}_2, i)\,. \tag{5.22}$$

We can thus bound the difference in the normalization terms $C_{1,i} = \sum_{k \in \mathcal{K}} \sigma(\mathbf{x}_i \mathbf{w}_{1,(k)})$ and $C_{2,i} = \sum_{k \in \mathcal{K}} \sigma(\mathbf{x}_i \mathbf{w}_{2,(k)})$. Since at most four classes can be affected by the change of two labels, we have:

$$C_{2,i} - 4\rho(\eta, \mathbf{W}_2, i) \leq C_{1,i} \leq C_{2,i} + 4\rho(\eta, \mathbf{W}_2, i)\,. \tag{5.23}$$

We can now bound the difference in the probabilities $p_{1,k,i}$ and $p_{2,k,i}$ as follows:

$$\begin{aligned} |p_{1,k,i} - p_{2,k,i}| &= \left| \frac{\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)})}{C_{2,i}} - \frac{\sigma(\mathbf{x}_i \mathbf{w}_{1,(k)})}{C_{1,i}} \right|, \\ &\leq \max\Big(\left| \frac{\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)})}{C_{2,i}} - \frac{\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)}) - \rho(\eta, \mathbf{W}_2, i)}{C_{2,i} + 4\rho(\eta, \mathbf{W}_2, i)} \right|, \\ &\qquad \left| \frac{\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)})}{C_{2,i}} - \frac{\sigma(\mathbf{x}_i \mathbf{w}_{2,(k)}) + \rho(\eta, \mathbf{W}_2, i)}{C_{2,i} - 4\rho(\eta, \mathbf{W}_2, i)} \right| \Big)\,. \end{aligned} \tag{5.24}$$

Taking the maximum of the right hand side of (5.24) over k leads to $|p_{1,k,i} - p_{2,k,i}| \leq \beta(\mathbf{W}_2, \eta, i)$ for all k. □

Bound on risk. Recall the definition of the risk of querying node q given a current label set $\mathbf{Y}_{\mathcal{L}_t}$, this time for multiclass classification:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} \triangleq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{U}_t^{-q}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} \mathbb{P}(y_q = k | \mathbf{Y}_{\mathcal{L}_t})\,, \tag{5.25}$$

where

$$\begin{aligned}\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} &\triangleq \Big(1 - \max_{k' \in \mathcal{K}} \mathbb{P}(y_i = k' | \mathbf{Y}_{\mathcal{L}_t}, y_q = k)\Big) \\ &= \min_{k' \in \mathcal{K}} \mathbb{P}(y_i = k' | \mathbf{Y}_{\mathcal{L}_t}, y_q = k).\end{aligned} \tag{5.26}$$

As for the binary case, the difference in risk is:

$$R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - \hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}} = \tfrac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}] - E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}'_{\mathcal{L}_t}}]\,, \tag{5.27}$$

The random variable $\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}$ now takes on value $\varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}}$ with probability $p(y_q = k|\mathbf{Y}_{\mathcal{L}_t})$ for $k \in \mathcal{K}$.

We can now state the following theorem that bounds the difference in risk for multiclass classification that arises from differing sets of labels.

Theorem 5.5.2. *Consider multiclass regression performed via repeated one-vs-all L^2-regularized logistic regression with regularization parameter λ to two labelled datasets $(\mathbf{X}, \mathbf{Y}_{\mathcal{L}_t})$ and $(\mathbf{X}, \mathbf{Y}'_{\mathcal{L}_t})$ that differ by one label, associated with the node q^*_{t-1}. Let $\mathbf{w}^k_{2,(k')}$ be the weight vector learned for class k' using label data $\mathbf{Y}'_{\mathcal{L}_t} \cup \{y_q = k\}$, and let $\mathbf{W}^k_2$ be the matrix with these vectors as columns, for $k' \in \mathcal{K}$. The risk error arising from using $\mathbf{Y}'_{\mathcal{L}_t}$ instead of $\mathbf{Y}_{\mathcal{L}_t}$ is bounded as:*

$$|R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - R^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}}| \leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \tilde{\beta}(\mathbf{W}_2, \eta_q, i)\,. \tag{5.28}$$

Here $\tilde{\beta}(\mathbf{W}_2, \eta_q, i) = \max_{k \in \mathcal{K}} \beta(\mathbf{W}^k_2, \eta_q, i)$ for $\beta(\mathbf{W}^k_2, \eta_q, i)$ as defined in (5.19).

Proof. For query node q, for each $k_1, k_2 \in \mathcal{K}$, we learn weights $\mathbf{w}^{k_1}_{2,(k)}$ using $\mathbf{Y}_{\mathcal{L}_t} \cup \{y_q{=}k_1\}$ and $\mathbf{w}^{k_2}_{2,(k)}$ using $\mathbf{Y}'_{\mathcal{L}_t} \cup \{y_q{=}k_2\}$, for each candidate class k. For each $i \in \mathcal{U}_t^{-q}$,

we have:

$$
\begin{aligned}
|\varphi^{+q}_{i,k_1,\mathbf{Y}_{\mathcal{L}_t}} - \varphi^{+q}_{i,k_2,\mathbf{Y}'_{\mathcal{L}_t}}| &= \left| \max_{k' \in \mathcal{K}} \mathbb{P}(y_i = k' | \mathbf{Y}'_{\mathcal{L}_t}, y_q = k_2) - \max_{k' \in \mathcal{K}} \mathbb{P}(y_i = k' | \mathbf{Y}_{\mathcal{L}_t}, y_q = k_1) \right| , \\
&\leq \max \left(\max_{k' \in \mathcal{K}} |\mathbb{P}(y_i = k' | \mathbf{Y}'_{\mathcal{L}_t}, y_q = k_2) - \mathbb{P}(y_i = k' | \mathbf{Y}_{\mathcal{L}_t}, y_q = k_1)| \right) , \\
&\leq \beta(\mathbf{W}_2^{k_2}, \eta_q, i) , \\
&\leq \max_{k_2 \in \mathcal{K}} \ \beta(\mathbf{W}_2^{k_2}, \eta_q, i) \triangleq \tilde{\beta}(\mathbf{W}_2, \eta_q, i) . \qquad (5.29)
\end{aligned}
$$

where the last line follows from Proposition 2.

Now, the difference in risk is bounded as

$$
\begin{aligned}
|R^{+q}_{|\mathbf{Y}_{\mathcal{L}_t}} - \hat{R}^{+q}_{|\mathbf{Y}'_{\mathcal{L}_t}}| &\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} |E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}_{\mathcal{L}_t}}] - E_{y_q}[\varphi^{+q}_{i,\mathbf{Y}'_{\mathcal{L}_t}}]| , \\
&\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \max \Bigg(|\max_{k \in \mathcal{K}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} - \min_{k \in \mathcal{K}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}| , \\
&\qquad\qquad |\min_{k \in \mathcal{K}} \varphi^{+q}_{i,k,\mathbf{Y}_{\mathcal{L}_t}} - \max_{k \in \mathcal{K}} \varphi^{+q}_{i,k,\mathbf{Y}'_{\mathcal{L}_t}}| \Bigg) , \text{ using Lemma (5.5.3)} , \\
&\leq \frac{1}{|\mathcal{U}_t^{-q}|} \sum_{i \in \mathcal{U}_t^{-q}} \tilde{\beta}(\mathbf{W}_2, \eta_q, i) , \text{ using (5.29)} . \qquad (5.30)
\end{aligned}
$$

□

5.5.4 Additional Lemmas

The proofs of the main results bounding the differences in risks rely on the following two lemmas:

Lemma 5.5.3. *Let X and Y be two random variables taking values in $[a, b]$ and $[c, d]$, respectively. Then $|E_X[X] - E_Y[Y]| \leq \max\Big(|a - d|, |b - c|\Big)$.*

Proof. We note

$$a - d \leq E_X[X] - E_Y[Y] \leq b - c\,,$$

from which the result follows. □

Lemma 5.5.4. *If* $0 \leq p_1, p_2 \leq 1$, *then* $|\min(p_1, 1-p_1) - \min(p_2, 1-p_2)| \leq |p_1 - p_2|$.

Proof. Let $q = \min(p_1, 1-p_1) - \min(p_2, 1-p_2)$.

$$\text{case } 1: \; p_1 < 0.5, \quad p_2 < 0.5$$

$$q = p_1 - p_2$$

$$\text{case } 2: \; p_1 \geq 0.5, \quad p_2 \geq 0.5$$

$$q = (1-p_1) - (1-p_2) = p_2 - p_1$$

$$\text{case } 3: \; p_1 < 0.5, \quad p_2 \geq 0.5$$

$$p_1 - p_2 \leq q = p_1 - (1-p_2) \leq (1-p_1) - (1-p_2) = p_2 - p_1$$

$$\text{case } 4: \; p_1 \geq 0.5, \quad p_2 < 0.5$$

$$p_2 - p_1 = (1-p_1) - (1-p_2) \leq q = (1-p_1) - p_2 \leq p_1 - p_2$$

So, we have $|q| \leq |p_1 - p_2|$. □

5.5.5 Discussion of the Bounds

In this section, bounds on the one-step risk differences were presented for both the binary and multiclass cases. A key feature of the bounds is that they are solely expressed in terms of the available weights $\mathbf{w}_2$ or $\mathbf{W}_2$, which means that each bound can be evaluated numerically (with minimal overhead compared to the regression

computations). If the bounds were tight enough, the algorithm could make an informed decision at the time of computation as to whether it would be beneficial to wait for the next label before determining which node to query. If the risk error is small enough such that there is no possibility of it impacting the query decision, then it is certain that the preemptive approximation will not affect the process at all. We do not explore the usage of the bound in this way in this thesis, but it could be an interesting direction for further study.

5.6 Summary

This chapter presented the preemptive algorithm PreGEEM that is based on approximating GEEM in order to perform computation before obtaining the label of the previous query. The purpose of this algorithm is to propose a way to take advantage of time that could potentially be wasted. Empirical results showed that the approximation did not hinder performance and a case study presented the effect of the approximation on the process by comparing it with GEEM. Lastly, bounds on the error made by approximation were presented for the binary and multiclass classification tasks.

CHAPTER 6
Conclusion

The work described in this thesis has contributed to the field of active learning for attributed graphs in multiple ways. First and foremost, a state-of-the-art algorithm for the problem of active learning for semi-supervised attributed node classification was introduced. In contrast to alternative approaches, the method relies on the principled framework of expected error minimization and uses graph-cognizant logistic regression to compete with GCN-based methods, without requiring a validation set. Second, an extension of this model was provided to make the algorithm applicable to the specific setting considered by the active learning for non-attributed graph literature. This variant, based on Bayesian model averaging, allows the algorithm to rely on a label propagation method in very early stages of the process and then transition to the better performing GEEM after enough labels have been collected. Third, a new problem formulation led to the development of the PreGEEM algorithm, another version of GEEM designed to perform preemptive querying by approximating ground truth labels with predictions. This is a time efficient approach because the alternative is to stall the process while waiting for the labelling process to be completed (assuming that the labelling process is time-consuming). The last contribution is to provide some theoretical guarantees for key quantities of the PreGEEM by bounding the error caused by the introduced approximation.

Numerical experiments demonstrated that the proposed method significantly outperforms existing active learning algorithms for attributed graphs, even when the competing algorithms are given the unfair advantage of having access to fine-tuned hyperparameters. This was thoroughly tested as the results were found to be consistent over four different benchmark datasets. The second and third experimental settings were conducted to evaluate how the designed combined algorithm compared to state-of-the-art algorithms for the non-attributed graph setting. The strength and consistency of our proposed model was once again highlighted as the reported results did not vary much while migrating from the transductive to the inductive case. By comparison, the label propagation methods designed for non-attributed graphs completely fail when applied to the inductive setting. In Chapter 5, which introduced the PreGEEM algorithm, the experiments focused on ensuring that performance did not deteriorate substantially due to the modifications and approximations introduced for the sake of time efficiency. The results demonstrate that in terms of achieved accuracy, PreGEEM is almost equivalent to its non-preemptive counterpart.

Future Work. There are multiple avenues and algorithmic extensions that can be explored. The main limitation of the proposed algorithms, which applies to any single node pool-based query strategy in general, is scalability. As the size of the graph grows, the relative impact of adding only one node decreases. This makes the approach less relevant. At the same time, the computational overhead increases. For this reason, we have limited our study to graphs with approximately 20,000 nodes. For graphs of such size, the proposed single node approaches are both computationally feasible and achieve noticeable advantages compared to random query node

selection. For graphs of larger size, the advantage diminishes. For larger graphs, a batch query solution is better suited, where multiple nodes can be selected at the same query step. One strategy that could be explored is to extend the proposed algorithm to the batch query setting. A simple, but heuristic and suboptimal approach, is just to identify multiple nodes at each step (the ten lowest risk estimates, for example). One could improve this by requiring that the selected nodes have predictions associated with different classes. Beyond such heuristics, it would be preferable to develop a more principled approach. Another possible direction would be to address the problem of applying deep learning algorithms to active learning scenarios more directly. The solution taken in this thesis was effectively to avoid the use of such models. A different approach could be to employ methods that focuses on training models on very few training instances such as few shot learning [42] or transfer learning [43].

REFERENCES

[1] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.

[2] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. on Learning Representations*, Toulon, France, Apr. 2017.

[3] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, US, Dec. 2017, pp. 1024–1034.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. on Learning Representations*, Vancouver, Canada, Apr. 2018.

[5] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proc. Int. World Wide Web Conf.*, Lyon, France, Apr. 2018, pp. 499–508.

[6] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. Int. Conf. on Knowledge Discovery & Data Mining*, London, United Kingdom, Aug. 2018, pp. 1416–1424.

[7] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, and L. Song, "Geniepath: Graph neural networks with adaptive receptive paths," in *Proc. AAAI Conf. on Artificial Intelligence*, Honolulu, HI, US, Jan. 2019, pp. 4424–4431.

[8] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Batch mode active learning and its application to medical image classification," in *Proc. Int. Conf. on Machine Learning*, Pittsburgh, PA, US, Jun. 2006, pp. 417–424.

[9] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proc. Int. Conf. on Machine Learning*, Sydney, Australia, Aug. 2017, pp. 1183–1192.

[10] T. Kurzendorfer, P. Fischer, N. Mirshahzadeh, T. Pohl, A. Brost, S. Steidl, and A. Maier, "Rapid interactive and intuitive segmentation of 3d medical images using radial basis function interpolation," *Journal of Imaging*, vol. 3, p. 56, Nov. 2017.

[11] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. on Machine Learning*, Washington, DC, USA, Aug. 2003, pp. 912–919.

[12] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella, "Active learning on trees and graphs," in *Proc. Conf. On Learning Theory*, Haifa, Israel, Jun. 2010.

[13] N. Cesa-Bianchi, "Active learning on graphs via spanning trees," in *Proc. Workshop on Networks Across Disciplines: Theories and Applications (NIPS)*, Vancouver, Canada, Jun. 2010.

[14] Q. Gu and J. Han, "Towards active learning on graphs: An error bound minimization approach," in *Proc. Int. Conf. on Data Mining*, Brussels, Belgium, Dec. 2012, pp. 882–887.

[15] M. Ji and J. Han, "A variance minimization criterion to active learning on graphs," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, La Palma, Canary Islands, Apr. 2012, pp. 556–564.

[16] Y. Ma, R. Garnett, and J. Schneider, "σ-optimality for active learning on Gaussian random fields," in *Proc. Advances in Neural Information Processing Systems*, Lake Tahoe, NV, US, Dec. 2013, pp. 2751–2759.

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

[18] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," in *Proc. Int. Conf. on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2019.

[19] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. on Neural Networks*, Québec, Canada, Jul. 2005, pp. 729–734.

[20] 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, 2009.

[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. on Learning Representations*, New Orleans, LA, US, Jun. 2019.

[22] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. EEEI Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, US, Jul. 2017, pp. 3693–3702.

[23] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. on Artificial Intelligence*, New Orleans, LA, US, Feb. 2018, pp. 629–638.

[24] M. Karzand and R. D. Nowak, "Active learning in the overparameterized and interpolating regime," *arXiv preprint arXiv:1905.12782*, 2019.

[25] 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, p. 93, Sep. 2008.

[26] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. Workshop on The Continuum from Labeled to Unlabeled Data (ICML)*, Washington, DA, US, Aug. 2003, pp. 58–65.

[27] A. Guillory and J. A. Bilmes, "Label selection on graphs," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2009, pp. 691–699.

[28] A. Guillory and J. Bilmes, "Active semi-supervised learning using submodular functions," in *Proc. Conf. on Uncertainty in Artificial Intelligence*, Barcelona, Spain, Jul. 2011, pp. 274–282.

[29] K. Jun and R. Nowak, "Graph-based active learning: A new look at expected error minimization," in *Proc. IEEE Global Conf. Signal and Information Proc.*, Greater Washington, DC, USA, Dec. 2016, pp. 1325–1329.

[30] D. Berberidis and G. B. Giannakis, "Data-adaptive active sampling for efficient graph-cognizant classification," *IEEE Trans. Signal Processing*, vol. 66, pp. 5167–5179, Oct. 2018.

[31] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *Proc. Int. Conf. on Machine Learning*, Haifa, Israel, Jun. 2010, pp. 79–86.

[32] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *Proc. germanInt. Conf. on Knowledge Discovery and Data Mining*, Las Vegas, NE, US, Aug. 2008, pp. 43–51.

[33] ——, "Reflect and correct: A misclassification prediction approach to active inference," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 4, Dec. 2009.

[34] J. Neville and D. Jensen, "Iterative Classification in Relational Data," in *Proc. Workshop on Learning Statistical Models From Relational Data (AAAI)*, Austin, TX, US, July 2000, pp. 42–49.

[35] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. on Machine Learning*, Long Beach, CA, US, Jun. 2019, pp. 6861–6871.

[36] T. Minka, "Bayesian model averaging is not model combination," 2002, MIT Media Lab Note.

[37] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: A tutorial," *Statistical Science*, vol. 14, no. 4, pp. 382–401, 1999.

[38] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Proc. Relational Representation Learning Workshop (NeurIPS)*, Montréal, Canada, Dec. 2018.

[39] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. Int. Conf. on Machine Learning*, San Francisco, CA, USA, June 2001, pp. 441–448.

[40] B. Settles, "From theories to queries: Active learning in practice," in *Proc. Active Learning and Experimental Design workshop In conjunction with AISTATS*, vol. 16, Sardinia, Italy, May 2011, pp. 1–18.

[41] H. Sivan, M. Gabel, and A. Schuster, "Online linear models for edge computing," in *Proc. Eur. Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Wurzburg, Germany, Sep. 2019.

[42] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *Proc. Int. Conf. on Learning Representations*, Vancouver, Canada, Apr. 2018.

[43] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artificial Neural Networks and Machine Learning*, Rhodes, Greece, Oct. 2018, pp. 207–217.