Dual Active Feature and Sample Selection for Graph Classification

Xiangnan Kong
University of Illinois at Chicago
Chicago, IL, USA
xkong4@uic.edu

Wei Fan
IBM T. J. Watson Research
Hawthorn, NY, USA
weifan@us.ibm.com

Philip S. Yu
University of Illinois at Chicago
Chicago, IL, USA
psyu@cs.uic.edu

ABSTRACT

Graph classification has become an important and active research topic in the last decade, where each instance is represented as a graph with complex structures. Current research on graph classification focuses on mining discriminative subgraph features under supervised settings. The basic assumption is that a large number of labeled graphs is available. However, labeling graph data is quite expensive and time-consuming in many real-world applications, such as testing molecular medicine's anti-cancer activities. In order to minimize the labeling efforts for subgraph mining in graph classification, we address the problem of how to actively select the most important graphs to obtain class labels where the objective is to mine the optimal subgraph features in order to construct accurate graph classifiers. This problem is challenging and different from previous works on active learning in that there are no predefined feature vectors and subgraph enumeration is NP-hard. In other words, the active sample selection and feature selection are "simultaneously correlated" problems for graph data. The simple reason is that before the most important graph can be estimated, a set of optimal subgraph features must be at hand. We demonstrate how to estimate the usefulness of a query graph and effectiveness of subgraph features at the same time, by maximizing the dependence between subgraph features and graph labels via a max-min view of pool-based active learning framework. Then we propose a branch-and-bound algorithm to efficiently search for optimal features and the optimal query graph simultaneously by judiciously pruning the subgraph search space. Empirical studies on real-world tasks demonstrate that our active feature and sample selection approach can obtain promising results for graph classification with much fewer (around 30% or even less) labeled graphs.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining

1. INTRODUCTION

With ever-increasing applications of data mining in disparate domains, graph data have become ubiquitous and increasingly important in modeling objects with complex structural information. Examples include chemical compounds, XML documents and program flows, *etc.* There is a great need for building models to automatically classify graph objects into different classes. For example, in chemoinformatics, researchers want to be able to predict the anti-cancer activities of chemical compounds in order to find new molecular drugs for cancers and chronic diseases [24, 15]; in computer software engineering, researchers are interested in studying how to identify errors/bugs in program flows automatically [5]. Motivated by these challenges, *graph classification* has received considerable attention in the last decade.

In the literature, graph classification problem has been extensively studied [16, 22, 3]. Conventional approaches focus on mining discriminative subgraph features [26] under supervised settings, which assume explicitly or implicitly that a large number of labeled graphs is available before the subgraph feature mining process. However, in many real-world applications, labeling graphs can be extremely expensive and time-consuming. For example, in molecular medicine, it requires time, efforts and excessive resources to test a drug's anti-cancer activities by pre-clinical studies and clinical trials; in software engineering, the domain experts need to examine an entire program flow carefully in order to decide whether there is an error in the program. The labeling cost for graph classification can be significantly reduced if one can train a model to actively select the most important graphs to query labels. This setting is also known as active learning or active query selection which aims to design models to exploit unlabeled data more effectively by iteratively selecting important examples to query, thus it can achieve comparable performances as supervised approaches while using much less labeled data. It has been shown useful in many real-world applications such as text classification [23, 27].

Formally, the active learning problem for graph data corresponds to minimizing the labeling cost by learning a model to actively select important graphs to obtain class labels in order to get promising performances on subgraph feature selection for graph classification. Active learning is particularly challenging in graph data. The reason is that, conventional active learning approaches can estimate the importance of each unlabeled examples in vector spaces while assuming all the useful features are given apriori before the

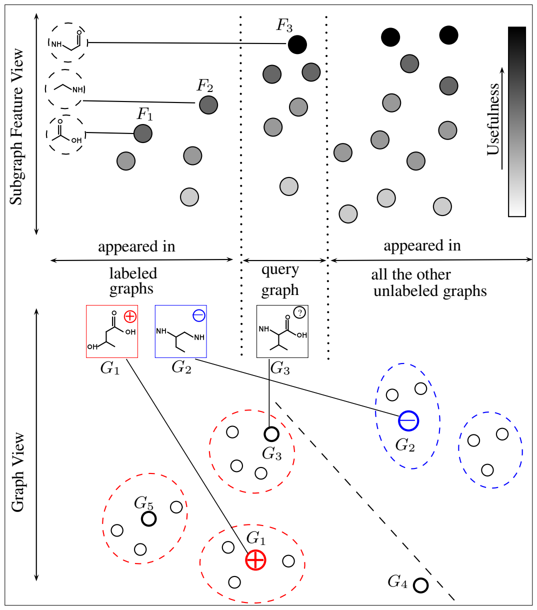


Figure 1: An Example of Dual Active Feature and Sample Selection in Graph Data.

iterative query selection process. But in graph data, the useful features are not given apriori, which require additional subgraph feature mining and selection by estimating the usefulness of subgraph features within all subgraph features in the graph dataset. What makes this problem even more interesting and challenging is that subgraph enumeration and graph isomorphism testing are NP-complete. Thus, it is impossible to enumerate all subgraph features and adopt existing approaches of active learning.

In active learning for graph data, the active query selection problem and the subgraph feature selection problem are closely related to each other. The reasons are as follows:

• In active query selection, we need to estimate the importance of each unlabeled graph in order to select the most important graphs to query labels. However, before the most important graphs can be estimated, a set of useful subgraph features must be at hand. Unlike conventional data in feature vector space, graph data are not directly represented in a meaningful feature space, and the active sample selection performance on graph data directly depends on the quality of the subgraph feature set we mined from the graph dataset. In other words, this is a chicken-egg problem. For example, suppose we aim to select the most important graphs to query labels like graph G_3 in Figure 1, which are both close to the class boundary (informative) like graph G_4 and representative to a cluster of unlabeled graphs like G_5. However, the informativeness and representativeness of a graph object depends on which feature set is used. The better feature set we use, the better we can estimate the importance of the query graphs among unlabeled graphs.

• In subgraph feature selection, we also need to select the most important subgraph features for the graph classification within a graph dataset. Conventional subgraph feature selection approaches for graph data focus on supervised settings [24, 15]. The feature evaluation strategies strictly follow the assumption that a large number of labeled graphs is given apriori. However, in active learning settings, we can only afford to query a small number of unlabeled graphs and

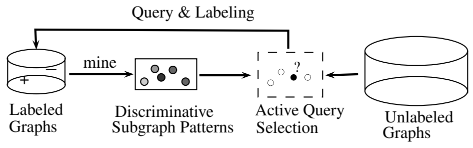


Figure 2: Two-stage Active Learning in Graph Data

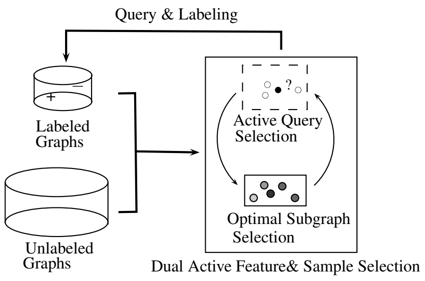


Figure 3: Dual Active Feature and Sample Selection

obtain their labels. The subgraph feature selection performance depends closely on the quality of the graphs which are queried in active sample selection process. For example, in Figure 1, suppose we are given two labeled graphs (G_1 and G_2), and only a small group of the useful subgraph features appeared in the labeled graphs like F_1 and F_2. If we select an unlabeled graph like G_3 to query, which is representative to a cluster of unlabeled graphs and far away from the two labeled graphs, we are more likely to find new useful features like F_3. From the feature selection view, the iterative query selection process is also a process to actively exploit useful features in the subgraph feature space. The better query graphs we select, the more effectively we can exploit the useful subgraph features.

Thus, the active sample selection and subgraph feature selection are correlated problems in graph data and should to be considered simultanously. Together they become a whole new problem, called “dual active feature and sample selection for graph classification”. *i.e.*, in order to minimize the labeling efforts for subgraph feature mining in graph classification, how to actively select the most important graphs to obtain class labels where the objective is to mine the most useful subgraph features in order to construct accurate graph classifiers.

Despite its value and significance, the dual active feature and sample selection for graph classification has not been studied in this context so far. One straight-forward solution to this problem would be the two-stage active learning framework for graph classification as shown in Figure 2. The feature selection and active sample selection are considered separately into two steps by iteratively selecting optimal subgraph features and selecting important query graphs. Obviously, in feature selection step, only the subgraph features that appeared in the label graphs can be found within this framework. And the estimation for the query graph in active sample selection may not be accurate since the useful features that only appeared in the unlabeled graphs are not able to be exploited in the feature selection step.

In this paper, we introduce a novel framework to the above problems by exploiting useful subgraph features and optimal query graph samples simultaneously. Our framework is illustrated in Figure 3. Different from the two-stage active learning method, the proposed approach, called gActive, can simultaneously estimate the usefulness of a query graph and

Table 1: Important Notations.

Symbol	Definition
$\mathcal{D} = \{G_1, \cdots, G_n\}$	the given graph dataset, G_i denotes the i-th graph in the dataset.
n_l, n_a and n_u	the number of labeled graphs, unlabeled graphs including and excluding the query graph in $\mathcal{D}$
$l = \{1, \cdots, n_l\}$	the index set for labeled graphs in $\mathcal{D}$
s and u	the index of the selected graph and the index set of the rest unlabeled candidate graphs in $\mathcal{D}$.
$a = \{n_l + 1, \cdots, n\}$	the index set for all unlabeled graphs in the pool including the selected graph. $a = \{s\} \cup u$
$\mathbf{y} = [y_1, \cdots, y_n]^\top$	the class label vector for graphs in $\mathcal{D}$, $y_i \in \{+1, -1, 0\}$
$\mathcal{S} = \{g_1, \cdots, g_m\}$	the set of all subgraph patterns in the graph dataset $\mathcal{D}$.
$\mathbf{x}_i = [x_i^1, \cdots, x_i^m]^\top$	the binary vector for G_i using subgraph features in $\mathcal{S}$, $x_i^k \in \{0, 1\}$ and $x_i^k = 1$ iff $g_k \subseteq G_i$
$\mathbf{f}_i = [f_i^1, \cdots, f_i^n]^\top$	the binary vector for subgraph pattern g_i in the $\mathcal{D}$
$X = [X_{ij}]_{(m \times n)}$	the matrix of all binary feature vectors in the dataset, $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] = [\mathbf{f}_1, \cdots, \mathbf{f}_m]^\top$
$\mathcal{T}$	the set of selected subgraph patterns, $\mathcal{T} \subset \mathcal{S}$
$\mathbf{K}(\mathcal{T}) = [K_{ij}]_{(n \times n)}$	the kernel matrix of all the graphs using the selected subgraph features $\mathcal{T}$
$\mathbf{L}(y_s, \mathbf{y}_l) = [L_{ij}]_{(n \times n)}$	the label kernel matrix of all the graphs based on the class labels
$\mathbf{H} = [H_{ij}]_{(n \times n)}$	the centering matrix, $H_{ij} = \delta_{ij} - n^{-1}$. ($\delta_{ij} = 1$ iff $i = j$, otherwise 0)
$D_{\mathcal{T}}$	an $m \times m$ diagonal matrix indicating which features are selected from $\mathcal{S}$ into $\mathcal{T}$
Π_l, Π_u and Π_s	mapping matrices, $\Pi_l \in \{0,1\}^{(n_l \times n)}$, $\Pi_s \in \{0,1\}^{(1 \times n)}$, $\Pi_u \in \{0,1\}^{(n_u \times n)}$ and $[\Pi_l^\top, \Pi_s^\top, \Pi_u^\top] = \mathbf{I}_n$

effectiveness of subgraph features by maximizing the dependence between subgraph features and graph labels based on a max-min view of pool-based active learning. Then we propose a branch-and-bound algorithm to efficiently search for optimal features and the query graph by judiciously pruning the subgraph search space. Empirical studies on real-world tasks demonstrate that our dual active feature and sample selection approach can obtain promising results in graph classification with much fewer labeled graphs.

2. RELATED WORK

To the best of our knowledge, this paper is the first work on dual active feature and sample selection problem for graph classification. Some research works have been done in related areas.

Active learning or active query selection deals with the problem of minimizing the labeling cost by designing active learner to choose which examples to label. Many works have been proposed base on various active learning settings (refer [21] for a detailed survey). Conventional active learning approaches focus on dealing with data in vector spaces, where it is assumed that all the useful features are available apriori and features should be fixed along the iterative labeling process. One well-know type of approaches is querying the most informative examples, where the active learner iteratively select the uncertain examples by the classifier [23, 1] or examples that has the largest disgreement among a committee of classifiers [6, 10, 20]. One problem with this type of approaches is that it is unable to exploit the structure among abundant unlabeled data and can be sensitive to outliers or noise in the dataset. Another type of active learning approaches is querying the most representative examples, where the active learners exploit the cluster structure of unlabeled data using clustering methods [18, 7] or optimal experimental design [28]. The main problem with this type of approaches is it is usually unsupervised and unable to make use of the labeling information from the labeled data. There are also works aim to combine the measure of informativeness and representiveness in vector space to find optimal query examples [8, 13].

Mining subgraph features from graph data have also been studied in recent years. The aim of such approaches is to extract useful subgraph features from a set of graphs by adopting some filtering criteria. Upon whether the label information is considered in the feature mining process, the existing works can roughly be classified into two types: unsupervised and supervised. In the unsupervised approaches, the frequencies are used as the subgraph feature evaluation criterion, where the aim is to collect frequently appearing subgraph features. For example, Yan and Han developed a depth-first search algorithm: gSpan [25], which can build a lexicographic order among graphs, and map each graph to an unique minimum DFS code as its canonical label. Based on the lexicographic order, gSpan algorithm adopts the depth-first search in the DFS code tree to mine frequent connected subgraphs efficiently. There are also many other frequent subgraph feature mining approaches have been developed in the last decade, *e.g.* AGM [14], FSG [17], MoFa [2], FFSM [12], and Gaston [19]. In the other hand, supervised subgraph feature mining approaches have also been proposed in the literature, such as LEAP [24], CORK [22], which search for discriminative subgraph features for graph classifications. In addition, gSSC [16] addresses the problem of feature selection for graph classification under semi-supervised settings.

3. PROBLEM FORMULATION

3.1 Dual Active Feature and Sample Selection

Consider a graph classificaiton problem where we are given a graph dataset $\mathcal{D} = \{G_1, \cdots, G_n\}$ that consists of n graphs. Let $\mathbf{y} = [y_1, \cdots, y_n]^\top$ denote the associated graph labels vector with $y_i \in \{+1, 0, -1\}$ where 0 implies that the graph is unlabeled. Active learning in graph data is the tasks of selecting one graph G_s from the pool of unlabeled graphs to query its labels. For convenience, we partition the graph dataset into three parts: the labeled graphs $\mathcal{D}_l$, the currently selected query graph G_s and the rest unlabeled graphs D_u. We also use $\mathcal{D}_a = \mathcal{D}_u \cup \{G_s\}$ to denote all the unlabeled graphs. The class labels vector $\mathbf{y}$ can also be partitioned accordingly,

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_l \\ y_s \\ \mathbf{y}_u \end{bmatrix} = \begin{bmatrix} \mathbf{y}_l \\ \mathbf{y}_a \end{bmatrix} \quad \text{and} \quad \mathbf{y}_a = \begin{bmatrix} y_s \\ \mathbf{y}_u \end{bmatrix}$$

where $\mathbf{y}_l$, y_s and $\mathbf{y}_u$ represent the class labels assigned to the graphs in $\mathcal{D}_l$, $\{G_s\}$ and $\mathcal{D}_u$ respectively. We denote n_l as the number of labeled graphs, n_u as the number of unlabeled graph except the query graph and n_a as the number of all the unlabeled graphs $n_a = n_u + 1$. Here we assume the first n_l graphs in the dataset are labeled.

Definition 1 (Graph). *A graph is represented as $G = (\mathcal{V}, E, \mathcal{L}, l)$, where $\mathcal{V}$ is a set of vertices $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\mathcal{L}$ is the set of labels for the vertices and the edges. $l : \mathcal{V} \cup E \rightarrow \mathcal{L}$, l is a function assigning labels to the vertices and the edges.*

We focus on using subgraph patterns to define the feature space of graph classification, which assumes that a graph object G_i is represented as a binary vector $\mathbf{x}_i = [x_i^1, \cdots, x_i^m]^\top$ associated with a set of subgraph patterns $\{g_1, \cdots, g_m\}$. Here $x_i^k \in \{0, 1\}$ is the binary feature of G_i corresponding to the subgraph pattern g_k, and $x_i^k = 1$ iff g_k is a subgraph of G_i (*i.e.* $g_k \subseteq G_i$). Now suppose the full set of subgraph features in the graph dataset $\mathcal{D}$ is $\mathcal{S} = \{g_1, \cdots, g_m\}$, which we use to predict the class labels of the graph objects. Usually the full feature set $\mathcal{S}$ is very large and only a subset of the subgraph features $\mathcal{T} \subseteq \mathcal{S}$ is relevant to the graph classification task. Let X denote the matrix consisting of the binary feature vectors using $\mathcal{S}$ to represent the graph dataset $\mathcal{D}$. $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_m]^\top \in \{0, 1\}^{m \times n}$, where $X = [X_{ij}]_{m \times n}$, $X_{ij} = 1$ iff $g_i \subseteq G_j$. We briefly summarize the notations in Table 1.

Definition 2 (Subgraph). *Let $G' = (\mathcal{V}', E', \mathcal{L}', l')$ and $G = (\mathcal{V}, E, \mathcal{L}, l)$ be graphs. G' is a subgraph of G ($G' \subseteq G$) iff there exist an injective function $\psi : \mathcal{V}' \rightarrow \mathcal{V}$ s.t. (1) $\forall v \in \mathcal{V}'$, $l'(v) = l(\psi(v))$; (2) $\forall (u, v) \in E'$, $(\psi(u), \psi(v)) \in E$ and $l'(u, v) = l(\psi(u), \psi(v))$. If G' is a subgraph of G, then G is a supergraph of G'.*

The key issue of dual active feature and sample selection for graph classification is how to simultaneously find the most important query graph from a pool of unlabeled graphs with a set of optimal subgraph patterns for graph classification. So, in this paper, the studied research problem can be described as follow:
1) How to estimate the importance of a query graph among unlabeled graphs combined with the optimal subgraph feature selection process?
2) How to properly evaluate the usefulness of a set of subgraph features based upon the labels of the graphs including the potential class label of the query graph?
3) How to determine the optimal subgraph features within a reasonable amount of time by avoiding the exhaustive enumeration based upon the labels of graphs?

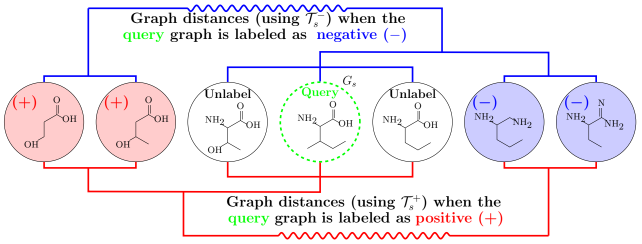
3.2 Optimization Framework

In this section, we address the problems discussed in Section 3.1 by defining the dual active feature and sample selection for graph classification as an optimization problem. The goal is to find an optimal query graph in a pool of unlabeled graphs together with feature selection process.

We propose the following general optimization framework to select the optimal query graph with a max-min view of pool-based active learning by maximizing the minimum score of an evaluation function.

$$G_s^* = \arg\max_{G_s \in \mathcal{D}_a} \min_{y_s \in \{\pm 1\}} \mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_l) \qquad (1)$$

where $\mathcal{D}_a$ denotes the pool of all the unlabeled graphs, and $\mathcal{E}$ denotes an evaluation function for querying a graph G_s in $\mathcal{D}_a$. As the label of the selected graph G_s is unkown, and hence can be either 1 or -1, we need to consider both alternatives and select the worse case to maximize. In this max-min view of active learning, it guarantees the selected



Graph distances (using $\mathcal{T}_s^-$) when the query graph is labeled as negative $(-)$
G_s
$(+)$
$(+)$
Unlabel
Query
Unlabel
$(-)$
$(-)$
O
OH
NH$_2$
N
Graph distances (using $\mathcal{T}_s^+$) when the query graph is labeled as positive $(+)$

Figure 4: Graph distances using two optimal feature sets ($\mathcal{T}_s^+$ and $\mathcal{T}_s^-$) respectively depending on the label of the query graph.

graph G_s will lead to a large value for the query selection evaluation function $\mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_l)$.

An important aspect of active learning for graph data is that the useful features are not given apriori, and we need to make use of the label information to select a subset of optimal subgraph features during the active sample selection process. Hypothetically, if we know the class label of a selected query graph G_s in addition to the labeled graphs in $\mathcal{D}_l$, we can select an optimal feature set by defining the query selection evaluation function as follow:

$$\mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_l) = \max_{\mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t} J(G_s, y_s, \mathcal{D}, \mathbf{y}_l, \mathcal{T}) \qquad (2)$$

where we select a subset of subgraph features $\mathcal{T}$ from $\mathcal{S}$, that can maximize a feature selection evaluation function $J(G_s, y_s, \mathcal{D}, \mathbf{y}_l, \mathcal{T})$.

3.2.1 *Intuitions*

For evaluation function, we assume that the optimal query graph together with its optimal subgraph feature sets should have the following properties:

Query Selection View: from the active query selection view, we assumes that (a) *Dependence Maximization*: based on a feature set $\mathcal{T}$, the query graph G_s should be able to maximize the dependence between features of the graphs and the labels in a max-min view as in Eq. 1. (b)*Informative and Representative:* the selected query graph should be both informative and representative among the pool of unlabel graphs. *i.e.* the query graph G_s should be close to some of the unlabeled graphs in the dataset, such that G_s is more likely to be representative of a group of unlabeled graphs instead of being outliers. In the other hand, the query graph G_s should also be far way from the labeled graphs in $\mathcal{D}_l$, such that the label of G_s is unlikely to be redundant.

Feature Selection View: from the subgraph feature selection view, we note that the optimal feature set can be very different depending on which graph we query and what class label we get from the domain expert. For example, in Figure 4, we have a graph dataset with seven graph objects. Suppose the query graph we selected is G_s, and we denote the optimal feature set as $\mathcal{T}_s^+$ when G_s is positive. $\mathcal{T}_s^-$ represents the optimal feature set when G_s is negative. From the feature selection perspective, the optimal subgraph feature set should also satisfy the following properties: (a) dependence maximization: graphs with the same class labels should have similar subgraph features, thus be close to each other; while graphs with different labels should have different features and far away from each other. (b) representativeness and informativeness: the query graph G_s should be close to the other unlabeled graphs and far away from the existing labeled graphs.

3.2.2 The Solution

According to a kernel-based dependence evaluation criterion name Hilbert-Schmidt Independence Criterion (HSIC) [11], we propose the following evaluation function for active feature selection:

$$J(G_s, y_s, \mathcal{D}, \mathbf{y}_l, \mathcal{T}) = \mathrm{tr}\left[\ \mathbf{K}(\mathcal{T})\ \mathbf{H}\ \mathbf{L}(y_s, \mathbf{y}_l)\ \mathbf{H}\ \right] + \alpha\frac{\mathbf{1}^\top\mathbf{K}_{u,s}(\mathcal{T})}{n_u} - \beta\frac{\mathbf{1}^\top\mathbf{K}_{l,s}(\mathcal{T})}{n_l} \tag{3}$$

where $\mathbf{K}(\mathcal{T}) = [K_{ij}]_{(n\times n)}$ denotes the kernel matrix of graphs based on a subgraph feature $\mathcal{T}$ and $K_{ij} = \langle D_\mathcal{T}\mathbf{x}_i, D_\mathcal{T}\mathbf{x}_j\rangle$. $D_\mathcal{T} = diag(\mathbf{d}_\mathcal{T})$ is a diagonal matrix indicating which features are selected into the feature set $\mathcal{T}$ from $\mathcal{S}$, $\mathbf{d}_\mathcal{T} = [d(\mathcal{T})_i]_{(m\times 1)}$ and $d(\mathcal{T})_i = I(g_i \in \mathcal{T}) \in \{0,1\}$. $\mathbf{L}(y_s, \mathbf{y}_l) = [L_{ij}]_{(n\times n)} = \mathbf{y}\mathbf{y}^\top$ denotes the graph kernel based on their class labels, where $\mathbf{y} = [\ \mathbf{y}_l^\top, y_s, \mathbf{y}_u^\top\ ]^\top$ and $L_{ij} = \langle y_i, y_j\rangle$ is used in our current implementation, other kernels can also be directly used in this formulation. α and β are two parameters, which control the weights of the three terms in the evaluation function. Here the first term denotes the dependence between subgraph features and the labels of graph objects. The second term represents the average similarities of the query graph selected to the other unlabeled graphs, and the third term represents the average similarities of the query graph to the labeled graph.

The evaluation function in Eq. 3 can be as follow:

$$\begin{aligned}\mathrm{tr}\Big(\mathbf{K}(\mathcal{T})\mathbf{H}\mathbf{y}\mathbf{y}^\top\mathbf{H}\Big) &= \mathrm{tr}\Big(X^\top D_\mathcal{T} D_\mathcal{T} X\ \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\Big)\\ &= \mathrm{tr}\Big(D_\mathcal{T} X\ \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ X^\top D_\mathcal{T}\Big)\\ &= \sum_{g_i\in\mathcal{T}}\Big(\mathbf{f}_i^\top\mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ \mathbf{f}_i\Big)\end{aligned}$$

and similarly we have:

$$\begin{aligned}&\alpha\frac{\mathbf{1}^\top\mathbf{K}_{u,s}(\mathcal{T})}{n_u} - \beta\frac{\mathbf{1}^\top\mathbf{K}_{l,s}(\mathcal{T})}{n_l}\\ =&\frac{\alpha}{n_u}\ \mathbf{1}^\top X_u^\top D_\mathcal{T} D_\mathcal{T}\mathbf{x}_s - \frac{\beta}{n_l}\ \mathbf{1}^\top X_l D_\mathcal{T} D_\mathcal{T}\mathbf{x}_s\\ =&\frac{\alpha}{n_u}\mathrm{tr}\Big[D_\mathcal{T}\mathbf{x}_s\mathbf{1}^\top X_u^\top D_\mathcal{T}\Big] - \frac{\beta}{n_l}\mathrm{tr}\Big[D_\mathcal{T}\mathbf{x}_s\mathbf{1}^\top X_l^\top D_\mathcal{T}\Big]\\ =&\frac{\alpha}{n_u}\mathrm{tr}\Big[D_\mathcal{T} X\Pi_s\mathbf{1}^\top\Pi_u^\top X^\top D_\mathcal{T}\Big] - \frac{\beta}{n_l}\mathrm{tr}\Big[D_\mathcal{T} X\Pi_s\mathbf{1}^\top\Pi_l^\top X^\top D_\mathcal{T}\Big]\\ =&\frac{\alpha}{n_u}\sum_{g_i\in\mathcal{T}}\Big(\mathbf{f}_i^\top\Pi_s\mathbf{1}^\top\Pi_u^\top\mathbf{f}_i\Big) - \frac{\beta}{n_l}\sum_{g_i\in\mathcal{T}}\Big(\mathbf{f}_i^\top\Pi_s\mathbf{1}^\top\Pi_l^\top\mathbf{f}_i\Big)\end{aligned}$$

where Π_l and Π_u are projection matrix as defined in Table 1, such that $X_l = X\Pi_l$, $X_u = X\Pi_u$ and $\mathbf{x}_s = X\Pi_s$

Now we can rewrite $J(G_s, y_s, \mathcal{D}, \mathbf{y}_l, \mathcal{T})$ in Eq. 2 as

$$\begin{aligned}&J(G_s, y_s, \mathcal{D}, \mathbf{y}_l, \mathcal{T})\\ =&\sum_{g_i\in\mathcal{T}}\mathbf{f}_i^\top\left(\mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_s\mathbf{1}^\top\Pi_u^\top - \frac{\beta}{n_l}\ \Pi_s\mathbf{1}^\top\Pi_l^\top\right)\mathbf{f}_i\\ =&\sum_{g_i\in\mathcal{T}}\mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i\end{aligned}$$

where we define

$$\mathbf{M} = \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_s\mathbf{1}^\top\Pi_u^\top - \frac{\beta}{n_l}\ \Pi_s\mathbf{1}^\top\Pi_l^\top \tag{4}$$

By defining an evaluation criterion $h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_l) = \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i$, the optimization for dual active feature and sample selection can be written as

$$G_s^* = \underset{G_s\in\mathcal{D}_a}{\arg\max}\ \min_{y_s\in\{\pm1\}}\ \max_{\mathcal{T}\subseteq\mathcal{S},|\mathcal{T}|=t}\sum_{g_i\in\mathcal{T}} h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_l) \tag{5}$$

Definition 3 (gFScore). *Let $\mathcal{D} = \{G_1, \cdots, G_n\}$ denote a graph dataset, with first n_l graphs labeled as $y_1, \cdots, y_{n_l}$. Suppose we have a query graph G_s with its potential label y_s, and $\mathbf{M}$ is a matrix defined as Eq. 4. We define a quality criterion $h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_l) = h(g_i, \mathbf{M}) = \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i$ called gFScore, for a subgraph feature g_i.*

The optimal solution to Eq. 5 can be found by using gFScore to mine the top-t subgraph feature sets for cases when each unlabeled graph is selected as query graph G_s and each case of the class label for G_s. As shown in Figure 6, we need to mine $2\times n_a$ optimal subgraph feature sets from the graph dataset, *i.e.* $\mathcal{T}_i^+$ denotes the optimal feature set when the i-th graph is selected as the query graph with the case when it is labeled as a positive graph; $\mathcal{T}_i^-$ denotes the optimal feature set when G_i is queried and labeled as negative. Then, we can directly use Eq. 5 to find the optimal graph to query.

3.2.3 Upper Bound of gFScore

Now we address the problem how to efficiently mine the optimal subgraph feature sets without exhaustive enumeration of all subgraph patterns in a graph dataset. Because the number subgraph patterns in graph dataset is usually extremely large, exponential to the size of the graphs, it is infeasible to enumerate each of the graphs and calculate its gFScore in order to find the top-t optimal subgraph features. Inspired by recent graph classification approaches [24, 22, 16], which put their feature evaluation criteria into the subgraph mining process and develop constraints to prune the subgraph search space in gSpan [25], we take a similar approach by deriving a different constraint to prune the pattern search space in the gSpan DFS code search tree.

A convenient method to compute an upper-bound on gFScore is given as follow:

Theorem 1 (Upper bound of gFScore). *Suppose we have two subgraph patterns $g_i, g_j \in \mathcal{S}$ and g_j is a supergraph of g_i ($g_j \supseteq g_i$). The gFScore value of g_j is bounded by $\widetilde{h}(g_i, \mathbf{M})$, i.e., $h(g_j, \mathbf{M}) \le \widetilde{h}(g_i, \mathbf{M})$. $\widetilde{h}(g_i, \mathbf{M})$ is defined as follow:*

$$\widetilde{h}(g_i, \mathbf{M}) \triangleq \mathbf{f}_i{}^\top\widetilde{\mathbf{M}}\mathbf{f}_i \tag{6}$$

where the matrix $\widetilde{\mathbf{M}}$ is defined as $\widetilde{M}_{pq} \triangleq \max(0, M_{pq})$.

Proof.

$$h\left(g_j, \mathbf{M}\right) = \mathbf{f}_j{}^\top\mathbf{M}\mathbf{f}_j = \sum_{p,q:G_p,G_q\in\mathcal{D}(g_j)} M_{pq} \tag{7}$$

where $\mathcal{D}(g_j) \triangleq \{G_k | g_j \subseteq G_k, 1 \le k \le n\}$. Since g_j is the supergraph of g_i ($g_j \supseteq g_i$), according to anti-monotonic property, we have $\mathcal{D}(g_j) \subseteq \mathcal{D}(g_i)$. Also $\widetilde{M}_{pq} \triangleq \max(0, M_{pq})$, we have $\widetilde{M}_{pq} \ge M_{pq}$ and $\widetilde{M}_{pq} \ge 0$. So,

$$\begin{aligned}h\left(g_j, \mathbf{M}\right) = &\sum_{p,q:G_p,G_q\in\mathcal{D}(g_j)} M_{pq} \le \sum_{p,q:G_p,G_q\in\mathcal{D}(g_j)} \widetilde{M}_{pq}\\ \le &\sum_{p,q:G_p,G_q\in\mathcal{D}(g_i)} \widetilde{M}_{pq} = \widetilde{h}\left(g_i, \mathbf{M}\right)\end{aligned} \tag{8}$$

Thus, for any $g_j \supseteq g_i$, $h(g_j, \mathbf{M}) \le \widetilde{h}(g_i, \mathbf{M})$. □

```
Input:
  $\mathcal{D}$: the graph dataset $\{G_1, \cdots, G_n\}$                    $t$: the maximum number of features.
  $\mathbf{y}_l$: the vector of class labels for labeled graphs,    $min\_sup$: the minimum frequency.
Initialize:
  - Construct the feature evaluation functions $h(g, \mathbf{M})$ and initialize candidate feature lists:
    1. Calculate $2 \times n_a$ matrices using Eq. 4 by considering each case for $G_s$ and $y_s$ as follow:
       $\mathbf{M}_i^+ = \mathbf{H}\ \mathbf{L}(y_i = +1, \mathbf{y}_l)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_l}\ \Pi_i \mathbf{1}^\top \Pi_l^\top$, $(\forall\ i,\ n_l < i \leq n)$
       $\mathbf{M}_i^- = \mathbf{H}\ \mathbf{L}(y_i = -1, \mathbf{y}_l)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_l}\ \Pi_i \mathbf{1}^\top \Pi_l^\top$, $(\forall\ i,\ n_l < i \leq n)$
    2. Initialize $2 \times n_a$ empty lists for candidate subgraph features as follow:
       $\forall\ i\ (n_l < i \leq n)$, let $\mathcal{T}_i^+ = \mathcal{T}_i^- = \emptyset$ with maximum size $t$, and pruning thresholds $\theta_i^+ = \theta_i^- = -\infty$
Recursive Features Mining:
  - Depth-First Search the gSpan's code tree and update the feature lists as follow:
    1. Update each of the candidate feature lists using the current subgraph feature $g_c$:
       $\forall\ i$, if $h(g_c, \mathbf{M}_i^+)$ is larger than the worst feature in $\mathcal{T}_i^+$, replace it and update $\theta_i^+ = \min_{g \in \mathcal{T}_i^+} h(g, \mathbf{M}_i^+)$
       $\forall\ i$, if $h(g_c, \mathbf{M}_i^-)$ is larger than the worst feature in $\mathcal{T}_i^-$, replace it and update $\theta_i^- = \min_{g \in \mathcal{T}_i^-} h(g, \mathbf{M}_i^-)$
    2. Test pruning criteria for the sub-tree rooted from node $g$ as follow:
       if $freq(g_c) < min\_sup$, prune the sub-tree of $g_c$
       if $\forall\ i\ (n_l < i \leq n)$, $\widetilde{h}(g_c, \mathbf{M}_i^+) \leq \theta_i^+$ and $\widetilde{h}(g_c, \mathbf{M}_i^-) \leq \theta_i^-$, prune the sub-tree of $g_c$
    3. Recursion: Depth-first search the sub-tree rooted from node $g_c$
Active Query Selection:
  - Select the query graph using Eq. 5
Output:
  $G_s$:    The selected query graph.
  $\mathcal{T}_s^+$:   the optimal subgraph feature set if $G_s$ is labeled as a positive graph.
  $\mathcal{T}_s^-$:   the optimal subgraph feature set if $G_s$ is labeled as a negative graph.
```

Figure 5: The gActive algorithm

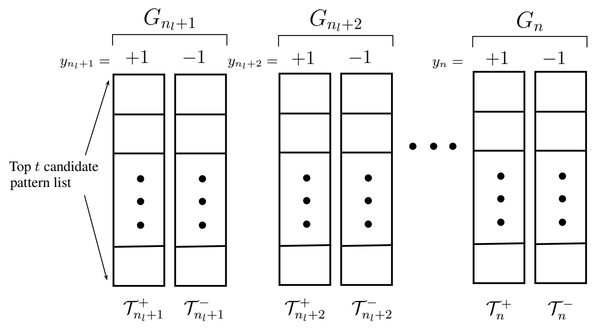


Figure 6: Candidate Subgraph Pattern Lists

We can now utilize the upper bound to efficiently prune the DFS Code Tree in gSpan by maintaining $2 \times n_a$ top-t best candidate feature lists as shown in Figure 6 with branch-and-bound pruning. During the course of subgraph pattern mining, we calculate the upper-bound of each subgraph pattern in the search tree. If the subgraph pattern node with its children nodes cannot update any of the candidate feature lists, we can prune the sub-tree of gSpan rooted from this node, and it is guaranteed by the upper-bound that we will not miss any better subgraph patterns for any of the candidate feature lists. Thus the subgraph feature mining process can speed up without loss of performance. The algorithm of gActive is summarized in Figure 5.

4. EXPERIMENTS

4.1 Experimental Setup

Data Collections: In order to evaluate the performances of our dual active feature and sample selection approach for graph classification, we tested our algorithm on nine real-world graph classification datasets as summarized in Table 2.

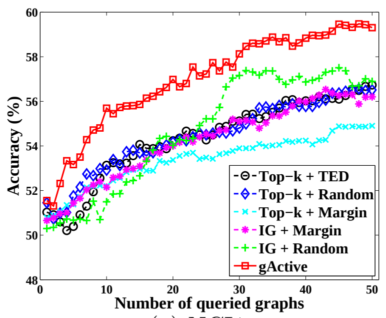
1) *Anti-cancer activity prediction (NCI)*: The first eight benchmark data sets are collected from PubChem Website[1]. The task is to classify chemical compounds' anti-cancer activities on each of the eight types of cancers, *e.g.* breast cancer, lung cancer and leukemia. The data sets consist information on the biological activities of small molecules, containing anti-cancer activity records of more then 20,000 chemical compounds against the eight types of cancers. Each chemical compound is represented as a graph. We collected 8 graph data sets with *active* and *inactive* labels from PubChem Website. The original datasets are unbalanced, where the active class is around 5%. We randomly sample 500 inactive compounds and 500 active compounds from each dataset for performance evaluation.

Table 2: Summary of experimental datasets. "# Pos" denotes the number of active graphs in the dataset.

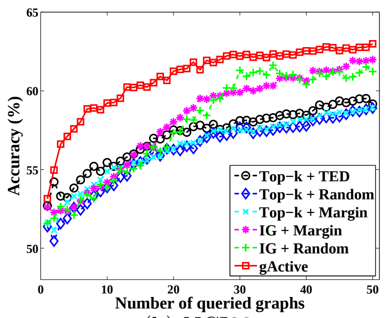
Dataset	# Pos	# Graph	Details
NCI1	2040	40526	Lung Cancer
NCI33	1636	40209	Melanoma
NCI41	1561	27585	Prostate Cancer
NCI47	2011	40447	Central Nerve System
NCI81	1396	40700	Colon Cancer
NCI83	2276	27992	Breast Cancer
NCI123	3112	40152	Leukemia
NCI145	1940	40164	Renal Cancer
AIDS	266	7781	HIV Anti-virus

2) *AIDS anti-virus prediction (HIV)*: The last benchmark dataset is collected from the AIDS anti-viral screen program[2]. The task is to classify chemical compounds' anti-virus activities on HIV. The data sets consist of screening records for more than 7700 chemical compounds. Each chemical compound is assigned with the screening result in

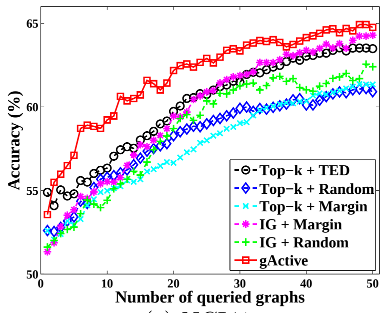
[1]http://pubchem.ncbi.nlm.nih.gov
[2]http://dtp.nci.nih.gov/



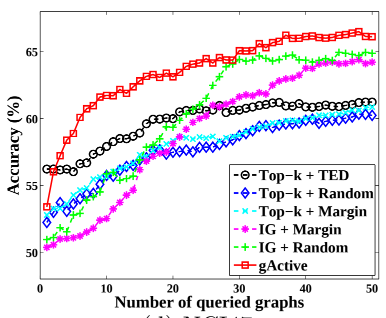
(a) NCI1



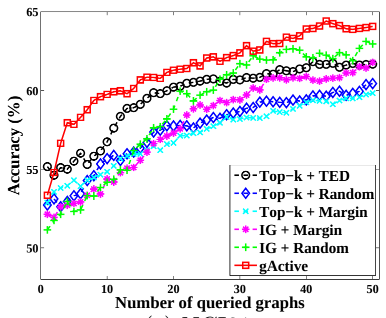
(b) NCI33



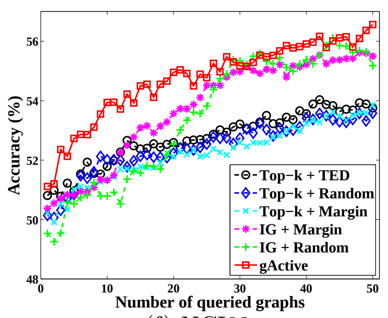
(c) NCI41



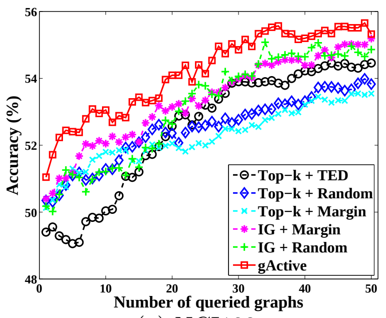
(d) NCI47



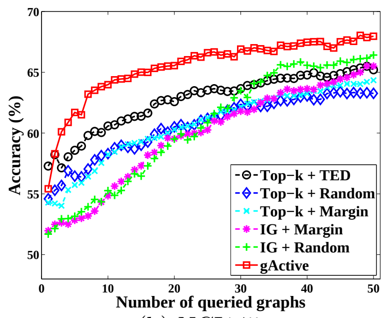
(e) NCI81



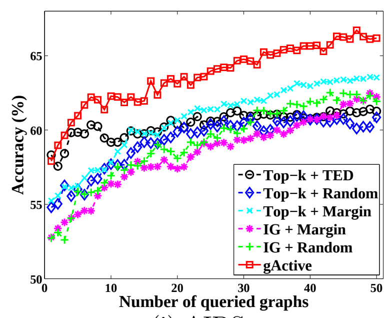
(f) NCI83



(g) NCI123



(h) NCI145



(i) AIDS

Figure 7: Graph classification accuracy after different number of graphs being queried.

one of the following three categories: confirmed active (CA), confirmed moderately active (CM) and confirmed inactive (CI). We assume CA+CM as *positive* labels, and CI as *negative*, which is the same setting as [9]. The original data sets are unbalanced, where the active class is around 3%. We randomly sampled 266 inactive compounds and used the original 266 active compounds for performance evaluation.

Comparing Methods: In order to demonstrate the effectiveness of our dual active feature and sample selection approach for graph classification, we compare our method with four baseline methods, including both supervised and unsupervised feature selection approaches combined with supervised and unsupervised active sample selection approaches. The compared methods are summarized as follows:

• *Dual Active Feature and Sample Selection (gActive)*: The proposed method in this paper which selects an important query graph at each iteration while outputting its corresponding optimal subgraph features. The parameters in gActive are set to $\alpha = \beta = 10^{-3}$ unless otherwise specified.

• *Supervised Feature Selection + Random Sampling (IG + Random)*: We compare with a supervised feature selection method with random sampling. A set of frequent subgraphs within the graph dataset are first mined. Then a supervised feature selection based upon Information Gain (IG), an entropy based measure, is used to select a subset of discriminative features from frequent subgraphs. We randomly select a query graph from the pool.

• *Supervised Feature Selection + Margin (IG + Margin)*: We compare with a supervised feature selection method with margin-based active learning for graph classification. In this approach, a set of frequent subgraphs within the graph dataset are first mined and we use information gain to select a subset of discriminative features from frequent subgraphs. Then margin-based active learning [23], a representative active learning approach is used to select informative graphs.

• *Unsupervised Feature Selection + Random Sampling (Top-k + Random)*: We also compare with an unsupervised feature selection method with random sampling. In this approach, we use the top-k frequent subgraph features in the pool dataset. Then we randomly select query graphs from the pool.

• *Unsupervised Feature Selection + Margin (Top-k + Margin)*: We also compare with an unsupervised feature selection method with margin-based active learning for graph classification. In this approach, we use the top-k frequent subgraph features in the pool dataset. Then margin-based active learning is used to select informative graphs to query.

- *Unsupervised Feature Selection + Sequential TED (Top-k + TED)*: We also compare with an unsupervised feature selection method based with an unsupervised active learning approach based on experiment design. In this approach, we use the top-k frequent subgraph features in the pool dataset. Then the sequential transductive experiment design approach [28] is used to select representative graphs from the pool dataset.

All experiments are conducted on machines with Intel Xeon^{TM} Quad-Core CPUs of 2.27 GHz and 24 GB RAM. LibSVM [4] with linear kernel is used as the base classifier for all the compared methods, and *min_sup* =10% in gSpan. The default number of selected features in all the compared methods is set as 500.

4.2 Performances on Graph Classification

In our experiments, we first randomly sample two labeled graphs from each dataset and used as initial training set, *i.e.* one positive and one negative graph. Then we partition rest of dataset into two parts with equal size: one part is used as the candidate pool data for active learning algorithms to query and the other part is used as test data set for performance evaluation. In each iteration, one unlabeled graph in the pool dataset is selected by each active learning method to be queried with its class label. Then classification models are retrained by incorporating the latest labeled graph. The results shown are the average of 50 runs on randomly sampled graph datasets.

The result of all the compared methods are shown in Figure 7. We show the number of queried graphs together with classification accuracies as the evaluation metric. Active learning methods iteratively select one graph to query class label in each iteration. We run each of the methods for 50 iterations and compare the learning curves.

In all these datasets, our dual active feature and sample selection algorithm (gActive) consistently outperforms other baseline methods on all the nine datasets. The most significant case is the NCI33 and NCI145, where both supervised and unsupervised feature selection with margin-based active learning baselines (IG+Margin and Top-k+Margin) are unable to improve the accuracy than random based baselines (IG+Random and Top-k+Random). Our method can still achieve substantially better performances. This result support our intuition that the feature selection and active query selection are correlated problems in graph data and should be optimized together in one framework. Moreover, we notice that in the dataset NCI41 when both the supervised and unsupervised feature selection with margin-based active learning baselines can improve the performance over random based baselines, our gActive method's improvements can be much more significant.

4.3 Parameter Settings

In our model we can take different weights on the three terms of the optimization in Eq. 3. If we use different settings for the two parameter α and β, we can perform the dual active feature and sample selection with different weights for the three types of constraints: *dependence maximization*, *representativeness* and *informativeness*. Here α represents how much we weight for the *representativeness*, and β denotes how much we weight the *informativeness*. The

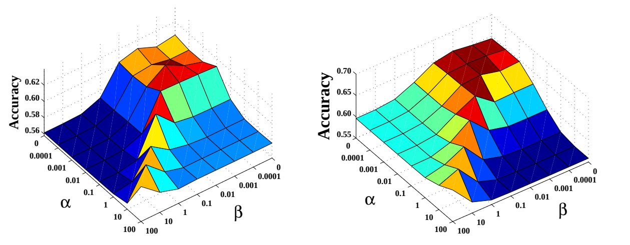


Figure 8: gActive accuracies with different α and β.

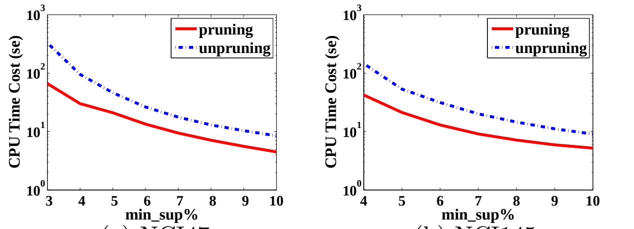


Figure 9: CPU time with/without pruning.

larger α is, the closer the query graph is with other unlabeled graphs. The larger the β is, the further away the query graph is from the other labeled graphs. We test α and β with values among {100, 10, 1, 0.1, 0.01, 0.001, 0} separately. The average results for our model in the first 50 iterations are reported. As shown in Figure 8, the performance of our model using α and β with similar values is often better than other settings. The reason is that in these real-world graph classification tasks, the constraint for *informativeness* and *representativeness* are equally important for our active feature and sample selection problem.

In Figure 8(a), we find that the best parameter setting for NCI47 dataset is $\alpha = 0.001$, $\beta = 0.001$ (accuracy = 63.5%), which is the same as our default parameter setting. The best parameter setting for NCI145 dataset is $\alpha = 0$, $\beta = 0.001$ (accuracy = 65.5%), and with our default parameter setting the accuracy is 65.4%. Generally, we can find that the performance of our gActive model with default setting is pretty good. If we try to optimize the selection of α and β value, the accuracy improvement over other baselines will be even bigger.

We also compared gActive models with and without pruning in the subgraph search space as shown in Figure 9. The average CPU time with different *min_sup* during the first iteration is reported. We can see that the gActive can improve the efficiencies by branch-and-bound search in the subgraph search space.

5. CONCLUSIONS

In this paper, we studied the problem of dual active feature and sample selection for graph classification in order to minimize the labeling efforts for feature selection in graph classification. We demonstrated how to simultaneously estimate the usefulness of a query graph and effectiveness of subgraph features candidates by maximizing the dependence between subgraph features and graph labels based on a max-min view of pool-based active learning. We propose to query the most representative and informative graph and select its corresponding optimal subgraph features. Then a branch-

and-bound algorithm is proposed to efficiently pruning the subgraph search space.

6. REFERENCES

[1] M. F. Balcan, A. Z. Broder, and T. Zhang. Margin based active learning. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 35–50, San Diego, CA, 2007.

[2] C. Borgelt and M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 211–218, Maebashi City, Japan, 2002.

[3] K. M. Borgwardt. *Graph Kernels*. PhD thesis, Ludwig-Maximilians-University Munich, 2007.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan. Identifying bug signatures using discrimative graph mining. In *Proceedings of the 18th International Symposium on Software Testing and Analysis*, pages 141–152, Chicago, IL, 2009.

[6] I. Dagan and S. P. Engenlson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning*, pages 150–157, Tahoe City, CA, 1995.

[7] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 208–215, Helsinki, Finland, 2008.

[8] P. Donmez, J. G. Carbonell, and P. N. Bennett. Dual strategy active learning. In *Proceedings of the 18th European Conference on Machine Learning*, pages 116–127, Warsaw, Poland, 2007.

[9] W. Fan, K. Zhang, H. Cheng, J. Gao, J. Han, P. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceedings of the 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 230–238, Las Vegas, NV, 2008.

[10] Y. Freund, E. S. H. S. Seung, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[11] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *ALT*, pages 63–77, Singapore, 2005.

[12] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 549–552, Melbourne, FL, 2003.

[13] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems 21*. 2011.

[14] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, Lyon, France, 2000.

[15] N. Jin, C. Young, and W. Wang. GAIA: graph classification using evolutionary computation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 879–890, Indianapolis, IN, 2010.

[16] X. Kong and P. S. Yu. Semi-supervised feature selection for graph classification. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 793–802, Washington, DC, 2010.

[17] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the 1st IEEE International Conference on Data Mining*, pages 313–320, San Jose, CA, 2001.

[18] H. T. Nguyen and A. W. M. . Smeulders. Active learning using pre-cluster. In *Proceedings of the 21th International Conference on Machine Learning*, pages 623–630, Banff, Canada, 2004.

[19] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 647–652, Seattle, WA, 2004.

[20] M. Opper, H. S. Seung, and H. Sompolinsky. Query by committee. In *ACM Workshop on Computational Learning Theory*, pages 287–294, Pittsburgh, PA, 1992.

[21] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[22] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. Yu, X. Yan, and K. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the 9th SIAM International Conference on Data Mining*, pages 1075–1086, Sparks, Nevada, 2009.

[23] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 999–1006, Stanford, CA, 2000.

[24] X. Yan, H. Cheng, J. Han, and P. Yu. Mining significant graph patterns by leap search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 433–444, Vancouver, BC, 2008.

[25] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 721–724, Maebashi City, Japan, 2002.

[26] X. Yan, P. S. Yu, and J. Han. Graph indexing based on discriminative frequent struture analysis. *ACM Transactions on Database Systems*, 30(4):960–993, 2005.

[27] B. Yang, J. Sun, T. Wang, and Z. Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 917–926, Paris, France, 2009.

[28] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *Proceedings of*

the 23rd International Conference on Machine Learning, pages 1081–1088, Pittsburgh, PA, 2006.