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. 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 are available. However, labeling graph data is quite expensive and time consuming for many real-world applications. In order to reduce the labeling cost for graph data, we address the problem of how to select the most important graph to query for the label. This problem is challenging and different from conventional active learning problems because there is no predefined feature vector. Moreover, the subgraph enumeration problem is NP-hard. The active sample selection problem and the feature selection problem are *correlated* for graph data. Before we can solve the active sample selection problem, we need to find a set of optimal subgraph features. To address this challenge, we demonstrate how one can simultaneously estimate the usefulness of a query graph and a set of subgraph features. The idea is to maximize the dependency between subgraph features and graph labels using an active learning framework. We propose a branch-and-bound algorithm to search for the optimal query graph and optimal features simultaneously. Empirical studies on nine real-world tasks demonstrate that the proposed method can obtain better accuracy on graph data than alternative approaches.

Categories and Subject Descriptors

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

General Terms

Algorithm, Experimentation, Performance

Keywords

Active Learning, Feature Selection, Graph Classification, Subgraph Pattern, Data Mining, Feature Construction

1. INTRODUCTION

Graphs arise naturally in a wide range of application domains. Examples include chemical compounds, XML documents and program flows. There is a great need for building models to classify graph objects. For example, in chemoinformatics, researchers would like to predict the anticancer activities of chemical compounds in order to find new drugs for cancer or chronic diseases [26, 17]; in software engineering, researchers are interested in studying how to automatically identify bugs in program flows [5]. Motivated by these challenges, *graph classification* has received considerable attention in the last decade.

Graph classification has been studied extensively in the literature [18, 24, 3]. Conventional approaches focus on mining discriminative subgraph features [28] under supervised settings. They assume explicitly or implicitly that a large number of labeled graphs are available. However, in many real-world applications, labeling graph data can be very expensive and time consuming. For example, in molecular medicine, it is very expensive to test the anticancer activity by preclinical studies and clinical trials; in software engineering, human experts have to examine a program flow carefully in order to find software bugs. The labeling cost for graph data can be significantly reduced by training a model that can select the most important graph to query for the label. This setting is also known as *active learning* or *active query selection.* It aims to exploit unlabeled data effectively by selecting important examples to query for labels. Thus, active learning approaches can usually achieve performance comparable to supervised approaches, while using less labeled data. Active learning has been shown to be useful in many real-world applications [25, 29].

Formally, the active learning problem for graph data corresponds to learning a model to select important graphs to obtain class labels. Active learning is particularly challenging in graph data. Conventional active learning approaches estimate the importance of unlabeled examples, and assume that all useful features are given. However, in graph data the useful features are not available. Thus, additional steps of subgraph feature mining and selection are required to estimate the usefulness of subgraphs. What makes this problem even more interesting and challenging is that subgraph enumeration and graph isomorphism testing problems are NP-

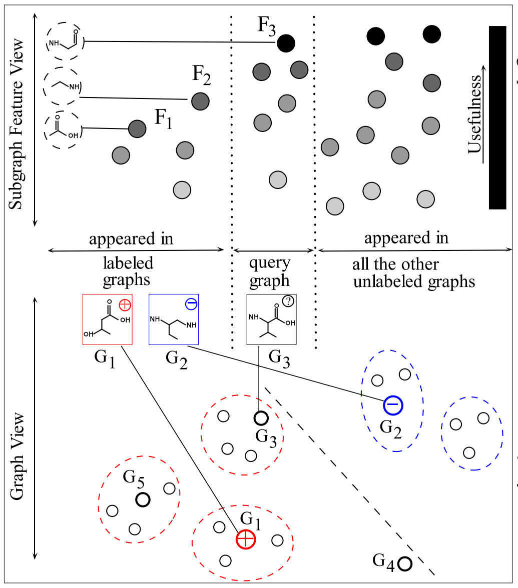


Figure 1: An example of dual active feature and sample selection in graph data.

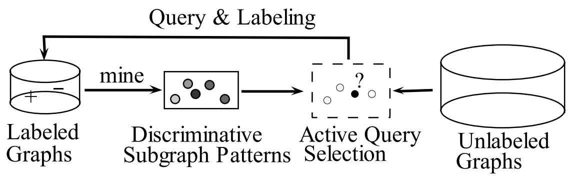


Figure 2: Two-stage active learning in graph data

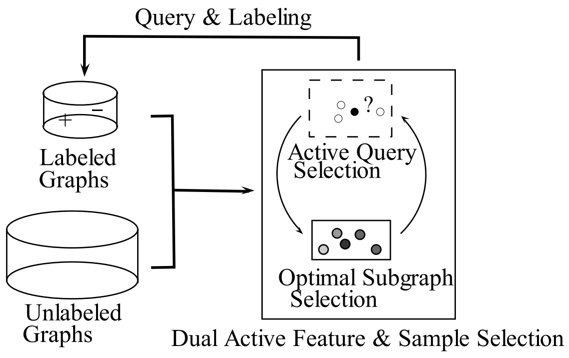


Figure 3: Dual active feature and sample selection

complete. Thus, it is impossible to enumerate all subgraph features and adopt existing approaches for 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 summarized as follows:

- In active query selection, we need to estimate the importance of each unlabeled graph in order to select the most important graph to query for the label. However, before the most important graph can be determined, we need to find a set of useful subgraph features. Graph data are not directly represented in a meaningful feature space. The performance of the active sample selection directly depends on the quality of the subgraph features mined from the graph dataset. For example, in Figure 1, G_3 is the most important graph which we want to query for the label. G_3 is close to the class boundary like graph G_4. Moreover, G_3 is representative of a cluster of unlabeled graphs. However, the informativeness and representativeness of a graph object depends on which feature set is used. The better the feature set we use, the better we will be able to estimate the importance of the query graphs.

- In the subgraph feature selection problem, we also need to select a set of important subgraph features for the graph classification task. Conventional feature selection approaches for graph data focus on supervised settings [26, 17]. The feature evaluation strategies strictly follow the assumption that a large number of labeled graphs are available. However, in the active learning settings, we can only afford to query a small number of unlabeled graphs and obtain their labels. The performance of the feature selection process depends strongly on the quality of the queried graphs in the active sample selection process. For example, in Figure 1, suppose we are given two labeled graphs (G_1 and G_2). Only a small number of the useful subgraph features (F_1 and F_2) appear in the labeled graphs. If we query the graph G_3 that is both representative and informative, we are more likely to find new features like F_3. The process of query selection can assist the process of feature selection in finding useful subgraph features. In other words, the better the query graph we select, the more effectively we can discover the useful subgraph features.

Thus, the active sample selection problem and the subgraph feature selection problem are correlated and should be considered simultaneously. The combined problem is referred to as *dual active feature and sample selection for graph classification*. This problem can be summarized as follows: in order to minimize the labeling cost in graph classification, we need to determine how one can actively select the most important graph to obtain the class label.

To the best of our knowledge, the dual active feature and sample selection for graph classification has not been studied in this context. A straightforward solution to this problem would be the two-stage active learning framework as shown in Figure 2. In this framework, the feature selection problem and the active sample selection problem are considered in two separate steps. In the first step, we select a set of subgraph features based upon the labeled graphs. In the second step, we estimate the importance of query graphs based upon the feature set selected in the first step. Obviously, only the subgraph features that appear in the label graphs can be found in the first step. The useful features that only appear in the unlabeled graphs can not be found. Thus, when we only use the features found in the first step, the estimation of the importance of query graphs will not be accurate.

In this paper we introduce a novel framework for the above problems by exploiting useful subgraph features and optimal query graphs simultaneously. Our framework is illustrated

Table 1: Important Notations.

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

in Figure 3. Unlike the two-stage active learning method, the proposed approach, called gActive, can estimate the usefulness of a query graph and effectiveness of subgraph features simultaneously. The gActive method maximizes the dependence between subgraph features and graph labels based upon an active learning framework. Furthermore, we propose a branch-and-bound algorithm to search for optimal features efficiently by pruning the subgraph search space. Empirical studies on real-world tasks demonstrate that the proposed method can obtain promising results using fewer labeled graphs than alternative approaches.

2. PROBLEM FORMULATION

Suppose we are given a graph dataset $\mathcal{D} = \{G_1, \cdots, G_n\}$ that consists of n graphs. $\mathbf{y} = [y_1, \cdots, y_n]^\top$ denotes the vector of labels, where $y_i \in \{+1, 0, -1\}$ is the label of G_i. $y_i = 0$ implies that G_i is unlabeled. Active learning in graph data is the task of selecting one graph G_s from the pool of unlabeled graphs to query for its label. For convenience, we partition the graph dataset into three parts: the labeled graphs $\mathcal{D}_\ell$, the query graph G_s, and the remaining unlabeled graphs D_u. $\mathcal{D}_a = \mathcal{D}_u \cup \{G_s\}$ denotes all unlabeled graphs. The vector $\mathbf{y}$ is partitioned as follows:

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

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

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

We focus on using subgraph patterns to define the feature space of graph classification. It is assumed 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. $x_i^k = 1$ iff g_k is a subgraph of 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. The full feature set $\mathcal{S}$ is very large. Only a subset of the features ($\mathcal{T} \subseteq \mathcal{S}$) is relevant to the graph classification task. Let X denote the matrix consisting of the binary feature vectors based on $\mathcal{S}$ 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 used in this paper 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 exists an injective function $\psi : \mathcal{V}' \to \mathcal{V}$. $\forall v \in \mathcal{V}'$, $l'(v) = l(\psi(v))$. $\forall (u,v) \in E'$, $(\psi(u), \psi(v)) \in E$ 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* is to find the most important query graph and a set of optimal subgraph patterns simultaneously. The problems studied in this paper are as follows:
1) How can one estimate the importance of a query graph among unlabeled graphs?
2) How can one estimate the usefulness of a set of features for the graph classification task?
3) How can one determine the optimal subgraph features within a reasonable amount of time, and avoid the exhaustive enumeration of all features?

2.1 Optimization Framework

We propose an optimization framework to select the optimal query graph by maximizing the minimum score of an evaluation function.

$$G_s^* = \underset{G_s \in \mathcal{D}_a}{\arg\max} \min_{y_s \in \{\pm 1\}} \mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) \quad (1)$$

where $\mathcal{D}_a$ denotes the pool of all unlabeled graphs. $\mathcal{E}$ denotes the evaluation function for querying a graph G_s in $\mathcal{D}_a$. Because the label of the selected graph G_s can be either 1 or -1, we need to consider both alternatives and select the worst case to maximize. In this max-min view of active learning, it guarantees that the selected graph G_s will lead to a large value for the function $\mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell)$.

Note that in graph data the useful features are not given. We need to make use of the label information to select a subset of optimal subgraph features. If we know the class label

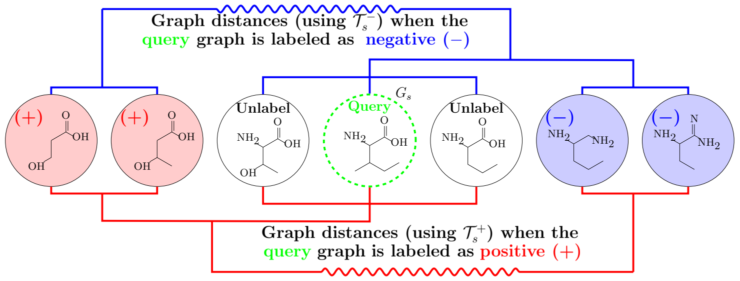

Graph distances (using $\mathcal{T}_s^-$) when the query graph is labeled as negative (−)
(+)
(+)
Unlabel
Query
G_s
Unlabel
(−)
(−)
Graph distances (using $\mathcal{T}_s^+$) when the query graph is labeled as positive (+)

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.

of the selected query graph G_s, we can select the optimal feature set by maximizing the following function.

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

We select a subset of subgraph features $\mathcal{T}$ from $\mathcal{S}$, such that the feature selection evaluation function $J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T})$ is maximized.

2.1.1 The Intuition

Now, we show the key ideas of this paper from the following views.

Query Selection View: from the active query selection view, we assume that the optimal query graph should satisfy the following properties:

(a) Dependence maximization: based upon a feature set $\mathcal{T}$, the query graph G_s should be able to maximize the dependence between features of graphs and 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 unlabeled graphs. The query graph G_s should be close to some of the unlabeled graphs in the dataset, such that G_s is likely to be representative of a group of unlabeled graphs instead of being an outlier. The query graph G_s should also be far from the labeled graphs in $\mathcal{D}_\ell$, such that the label of G_s is unlikely to be redundant.

Feature Selection View: from the subgraph feature selection view, 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 selected query graph is G_s. 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 features should also satisfy the following properties:

(a) Dependence maximization: graphs with the same class labels should have similar subgraph features, and be close to each other; while graphs with different labels should have different features and be far away from each other.

(b) Informative and representative: the query graph G_s should be close to the other unlabeled graphs, and be far from the existing labeled graphs.

2.1.2 The Solution

Many criteria can be used to evaluate the dependence between subgraph features and graph labels. In this paper we adopt the Hilbert-Schmidt Independence Criterion (HSIC) [12] for subgraph evaluation. HSIC measures the dependence between two variables (X, Y) in the kernel space. We briefly review the definition of HSIC. Suppose we have two reproducing kernel Hilbert spaces (RKHS) of functions $\mathcal{G}$ and $\mathcal{F}$. Let a covariance operator be

$$C = \mathrm{E}\left\{[p(X) - \mathrm{E}(p(X))][q(Y) - \mathrm{E}(q(Y))]\right\}, \ \ \forall p \in \mathcal{G}, q \in \mathcal{F}$$

Then the HSIC is defined as the Hilbert-Schmidt norm of the operator C, *i.e.*, $\|C\|^2_{HS}$. Given a data sample, HSIC has an empirical estimator HSIC $= \mathrm{tr}(\mathbf{K}\ \mathbf{H}\ \mathbf{L}\ \mathbf{H})$. Here, $\mathrm{tr}(\cdot)$ is the trace of a matrix. $\mathbf{H} = [H_{ij}]_{n\times n}$, where $H_{ij} = \delta_{ij} - 1/n$. δ_{ij} is the indicator function. $\delta_{ij} = 1$ iff $i = j$, otherwise $\delta_{ij} = 0$. $\mathbf{K}$ and $\mathbf{L}$ are kernel matrices on the samples.

Based upon the HSIC measure, we propose the following evaluation function for active feature selection:

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

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

The evaluation function in Eq. 3 can be simplified as follows:

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

```
Input:
  $\mathcal{D}$: the graph dataset $\{G_1, \cdots, G_n\}$          $t$: the maximum number of features.
  $\mathbf{y}_\ell$: 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 follows:
        $\mathbf{M}_i^+ = \mathbf{H}\ \mathbf{L}(y_i = +1, \mathbf{y}_\ell)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_i \mathbf{1}^\top \Pi_\ell^\top$, $(\forall\ i,\ n_\ell < i \le n)$
        $\mathbf{M}_i^- = \mathbf{H}\ \mathbf{L}(y_i = -1, \mathbf{y}_\ell)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_i \mathbf{1}^\top \Pi_\ell^\top$, $(\forall\ i,\ n_\ell < i \le n)$
    2. Initialize $2 \times n_a$ empty lists for candidate subgraph features as follows:
        $\forall\ i\ (n_\ell < i \le n)$, let $\mathcal{T}_i^+ = \mathcal{T}_i^- = \emptyset$ with maximum size $t$, and pruning thresholds $\theta_i^+ = \theta_i^- = -\infty$
Recursive Feature Mining:
  - Depth-first search the gSpan's code tree and update the feature lists as follows:
    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 follows:
        if $freq(g_c) < min\_sup$, prune the sub-tree of $g_c$
        if $\forall\ i\ (n_\ell < i \le n)$, $\widetilde{h}(g_c, \mathbf{M}_i^+) \le \theta_i^+$ and $\widetilde{h}(g_c, \mathbf{M}_i^-) \le \theta_i^-$, 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 features if $G_s$ is labeled as a positive graph.
  $\mathcal{T}_s^-$:   the optimal subgraph features if $G_s$ is labeled as a negative graph.
```

Figure 5: The gActive algorithm

We can rewrite $J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T})$ in Eq. 2 as

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

where Π_ℓ and Π_u are projection matrices. $X_\ell = X\Pi_\ell$, $X_u = X\Pi_u$ and $\mathbf{x}_s = X\Pi_s$.

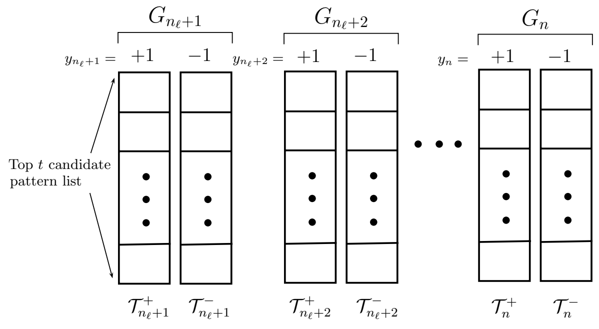
$$
\mathbf{M} = \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_s \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_s \mathbf{1}^\top \Pi_\ell^\top \qquad (4)
$$

Given the evaluation function $h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) = \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i$, we can define the optimization problem of dual active feature and sample selection as follows:

$$
G_s^* = \arg\max_{G_s \in \mathcal{D}_a}\ \min_{y_s \in \{\pm 1\}}\ \max_{\mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t} \sum_{g_i \in \mathcal{T}} h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) \qquad (5)
$$

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

The optimal solution to Eq. 5 can be found by the following method: we go through each unlabeled graph one by one. Suppose G_s is the currently selected graph. We need to go through a positive case ($y_s = 1$) and a negative case ($y_s = -1$). In each case, we mine the top-t best subgraph features according to the gScore. As shown in Figure 6, we need to mine $2 \times n_a$ optimal feature sets from the graph dataset. $\mathcal{T}_i^+$ denotes the optimal feature set, when the i-th



$G_{n_\ell+1}$
$G_{n_\ell+2}$
G_n
$y_{n_\ell+1} =$
$+1$
-1
$y_{n_\ell+2} =$
$+1$
-1
$y_n =$
$+1$
-1
Top t candidate pattern list
$\mathcal{T}_{n_\ell+1}^+$
$\mathcal{T}_{n_\ell+1}^-$
$\mathcal{T}_{n_\ell+2}^+$
$\mathcal{T}_{n_\ell+2}^-$
$\mathcal{T}_n^+$
$\mathcal{T}_n^-$

Figure 6: Candidate subgraph pattern lists

graph is queried and 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 query graph.

2.1.3 Upper Bound of gFScore

Now we address the problem on how to mine the optimal feature sets without exhaustive enumeration of all subgraphs. Note that the number of subgraphs in a graph dataset can be extremely large. It grows at an exponential rate as the size of the graphs increases. Thus, it is computationally intractable to enumerate all subgraphs in the graph dataset.

Some recent graph classification approaches [26, 24, 18] incorporate constraints to prune the search space of gSpan [27]. In this paper we derive a new constraint to prune the search space in the DFS-code tree. A straightforward upper-bound of gFScore is defined as follows:

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., $h(g_j, \mathbf{M}) \leq \widetilde{h}(g_i, \mathbf{M})$. $\widetilde{h}(g_i, \mathbf{M})$ is defined as follows:

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

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

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

For any $g_j \supseteq g_i$, $h(g_j, \mathbf{M}) \leq \widetilde{h}(g_i, \mathbf{M})$. □

We now utilize the upper bound to prune the DFS-code tree in gSpan by the branch-and-bound pruning. The top-t best features are maintained in $2 \times n_a$ candidate lists. Figure 6 shows an example of the candidate lists. During the course of the subgraph pattern mining, we calculate the upper-bound of each subgraph pattern in the search tree. If a subgraph pattern node with its children nodes cannot update any of the candidate feature lists, we can prune the subtree of gSpan rooted from this node. It is guaranteed by the upper-bound that we will not miss any better features for any of the candidate feature lists. Thus, the subgraph feature mining process can be speeded up without loss of performance. The algorithm of gActive is summarized in Figure 5.

3. EXPERIMENTS

Data Collection: in order to evaluate the performance of the proposed approach for graph classification, we tested our algorithm on nine real-world datasets as summarized in Table 2.

1) *Anti-cancer activity prediction (NCI)*: the first eight benchmark datasets are collected from the *PubChem* website[1]. The datasets contain records of anticancer activities for more than 20,000 chemical compounds against eight types of cancer. Each chemical compound is represented as a graph. We collected eight graph data sets with *active* and *inactive* labels from the *PubChem* website. The original datasets are unbalanced, where the percentage of positive compounds is around 5%. We randomly sampled 500 inactive compounds and 500 active compounds from each dataset for performance evaluation.

2) *AIDS anti-virus prediction (HIV)*: the last benchmark dataset is collected from the AIDS anti-viral screening program[2]. The dataset consists of screening records of more than 7700 chemical compounds. Each compound is described by its activity against HIV, which is one of the following categories: confirmed active (CA), confirmed moderately active (CM) and confirmed inactive (CI). We treat CA+CM as the *positive* label, and CI as the *negative* label. This is the same setting used in [9]. The original data set is unbalanced, where the percentage of positive compounds is around 3%. We randomly sampled 266 inactive compounds and used the original 266 active compounds for performance evaluation.

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

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

Comparative Methods: in order to demonstrate the effectiveness of the proposed approach, we compare our method against four baseline methods. These baseline methods include both supervised and unsupervised feature selection approaches combined with active sample selection approaches. The compared methods are summarized as follows:

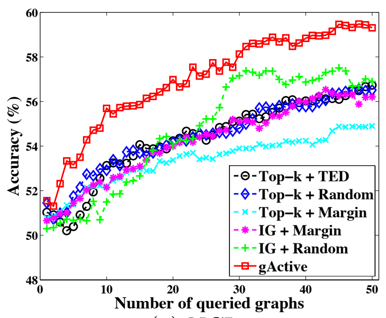
- *Dual Active Feature and Sample Selection (gActive)*: the proposed method in this paper. The default parameter setting for the gActive method is $\alpha = \beta = 10^{-3}$.

- *Supervised Feature Selection + Random Sampling (IG + Random)*: we compare with a supervised feature selection method combined with random sampling. A set of frequent subgraphs within the graph dataset is first mined. Then a supervised feature selection method based upon Information Gain (IG) is used to select a subset of discriminative features from the frequent subgraphs. We randomly select query graphs from the pool. Note that discriminative features are recomputed after each iteration of feature and graph selection.

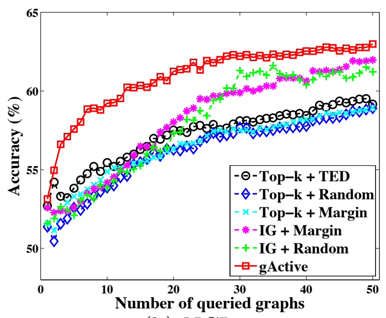
- *Supervised Feature Selection + Margin (IG + Margin)*: we compare with a two-stage active learning approach. A supervised feature selection method is combined with a margin-based active learning method. In this approach, a set of frequent subgraphs within the graph dataset are first mined. Then we iteratively use information gain to select a subset of discriminative features from the frequent subgraphs. We use a margin-based active learning approach [25] to select informative graphs to query for labels.

- *Unsupervised Feature Selection + Random Sampling (Top-k + Random)*: we compare with an unsupervised feature selection method combined 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.

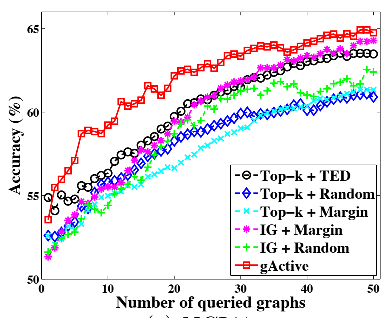
- *Unsupervised Feature Selection + Margin (Top-k + Margin)*: we compare with an unsupervised feature selection method combined with a margin-based active



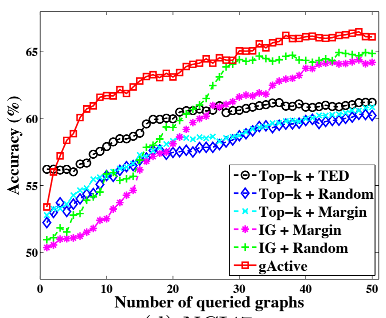
(a) NCI1



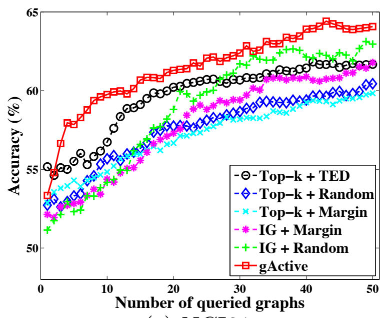
(b) NCI33



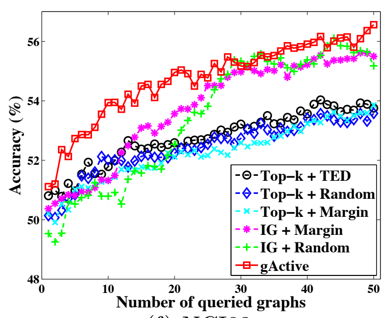
(c) NCI41



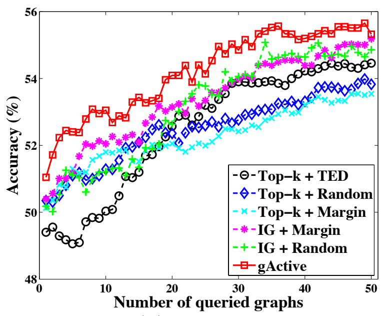
(d) NCI47



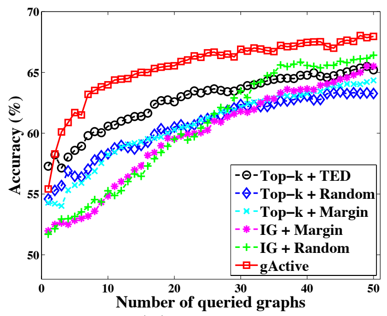
(e) NCI81



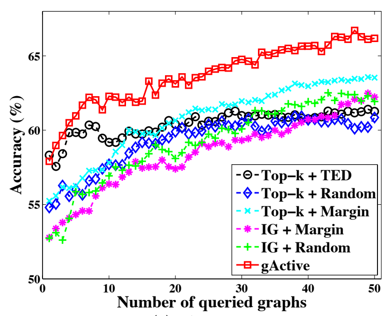
(f) NCI83



(g) NCI123



(h) NCI145



(i) AIDS

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

learning method. In this approach, we use the top-k most frequent subgraphs 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 compare with an unsupervised feature selection method combined with experimental designs. In this approach, we use the top-k most frequent subgraphs in the pool dataset. Then the sequential transductive experimental design [30] is used to select representative graphs from the pool dataset.

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

3.1 Performance on Graph Classification

In our experiments, we first randomly sample two labeled graphs, *i.e.*, one positive and one negative graph. They are used as the initial training set. Then we partition the remaining graphs into two groups with equal size: one group is used as the pool dataset; the other group is used as the testing dataset for performance evaluation. In each iteration, the active learner selects one unlabeled graph in the pool dataset and queries the label. Then, the queried graph is put into the training set, and the base classifier is trained to classify the testing graphs. We report the average results of 50 runs on randomly sampled graph datasets.

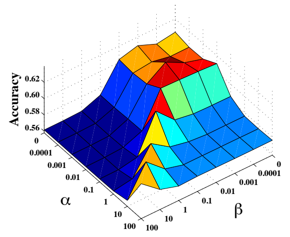
The results of all compared methods are summarized in Figure 7. We run each of the methods for 50 iterations and compare the learning curves. In all datasets, the proposed dual active feature and sample selection algorithm (gActive) consistently outperforms other baseline methods. Note that in the NCI33 and NCI145 datasets, all the baselines with margin-based active learning (IG + Margin and Top-k + Margin) are unable to achieve better performance than randomized baselines (IG + Random and Top-k + Random). However, our method can achieve substantially better performance than other baselines. This result supports the intuition of this paper: the feature selection problem and the active query selection problem are correlated, and should be considered simultaneously. Moreover, we notice that in the NCI41 dataset, all the baselines using margin-based active learning can improve the performance over randomized baselines. In this dataset, the improvements by the proposed gActive method can be even more significant.

3.2 Parameter Settings

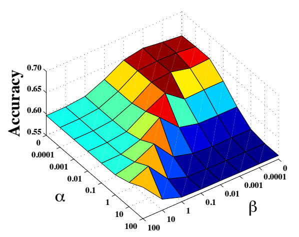
In the proposed model, different weights can be assigned to the three terms in Equation 3. If we use different settings for the two parameters, α and β, we can perform the dual active feature and sample selection with different weights for the three constraints: *dependence maximization*, *representativeness* and *informativeness*. To be precise, α represents how much we weight the *representativeness*, and β denotes how much we weight the *informativeness*. The larger the value of α is, the closer the query graph is with other unlabeled graphs. The larger the value of 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. In these real-world graph classification tasks, the constraints 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 $\alpha = 0.001$, $\beta = 0.001$, and the accuracy is 63.5%. This setting is the same as our default parameter setting. The best parameter setting for NCI145 dataset is $\alpha = 0$, $\beta = 0.001$, and the resulting accuracy is 65.5%. Under our default setting, the accuracy is 65.4%. We find that the performance of our gActive model with the default setting is satisfactory. If we try to optimize the selection of α and β values, the accuracy improvement over other baselines will be even larger.

We also compare gActive models with and without pruning in the subgraph search space as summarized in Figure 9. The average CPU time with different *min_sup* during the first iteration is reported. gActive can improve the efficiencies by pruning the subgraph search space.

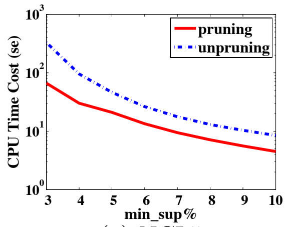


(a) NCI47

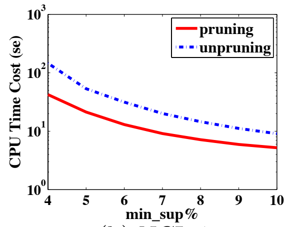


(b) NCI145

Figure 8: gActive accuracies with different α and β.



(a) NCI47



(b) NCI145

Figure 9: CPU time with/without pruning.

4. RELATED WORK

In this section, we discuss some related work. Active learning aims at reducing the labeling cost by querying the most informative example. Many methods have been proposed based upon different active learning settings. Please see [23] for a detailed survey. Conventional active learning approaches focus on data in vector space. One approach is to query the most informative instance. The active learners select the uncertain instances based upon a single classifier [25, 1] or a committee of classifiers [6, 10, 22]. The problems with this approach are that it can be sensitive to outliers or noise, and can not exploit the structures of unlabeled data. The alternative approach is to query the most representative instance. The active learners exploit the structure of unlabeled data using clustering methods [20, 7] or optimal experimental design approaches [30]. The major problems are that this approach is unsupervised and can not use the labeled data. Some work has also been done to combine informativeness and representativeness measures to find the optimal query examples [8, 15].

Mining subgraph features from graph data has also been studied in recent years [19]. The aim of these approaches is to extract useful subgraph features from a set of graphs by adopting some filtering criteria. Depending on if the label information is considered in the feature mining process, the existing methods can roughly be classified into two types: unsupervised and supervised. In the unsupervised setting, the frequency of each subgraph is used as the criterion for feature selection. To discover frequent subgraphs, Yan and Han developed a depth-first search algorithm, gSpan [27], which builds a lexicographic order among graphs, and maps each graph to a unique minimum DFS code as its canonical label. The gSpan algorithm adopts a depth-first search in the DFS code tree to mine frequent subgraphs efficiently. There are also many other frequent subgraph feature mining approaches that have been developed recently, *e.g.*, AGM [16], FSG [19], MoFa [2], FFSM [14], and Gaston [21]. Supervised subgraph feature mining approaches have also been proposed in the literature, such as LEAP [26], CORK [24] and MbT [9], where the focus is on discriminative features for graph classifications. In addition, gSSC [18] addresses the problem of feature selection for graph classification under semi-supervised settings. Some other graph classification approaches based upon graph kernels have also been proposed [11, 13].

5. CONCLUSIONS

In this paper we studied the dual active feature and sample selection problem for graph classification. The objective was to minimize the labeling efforts in graph classification. We demonstrated how to find a useful query graph and a set of optimal features simultaneously. We proposed to maximize the dependence between subgraph features and labels based upon an active learning framework. Our approach can find the most representative and informative graph and an optimal feature set. Then a branch-and-bound algorithm was proposed to prune the subgraph search space efficiently.

6. ACKNOWLEDGEMENTS

The work is supported in part by NSF through grants IIS-0905215, DBI-0960443, OISE-0968341 and OIA-0963278.

7. REFERENCES

[1] M. F. Balcan, A. Z. Broder, and T. Zhang. Margin based active learning. In *COLT*, pages 35–50, 2007.

[2] C. Borgelt and M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *ICDM*, pages 211–218, 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 *ISSTA*, pages 141–152, 2009.

[6] I. Dagan and S. P. Engenlson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pages 150–157, 1995.

[7] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215, 2008.

[8] P. Donmez, J. G. Carbonell, and P. N. Bennett. Dual strategy active learning. In *ECML*, pages 116–127, 2007.

[9] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *KDD*, pages 230–238, 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] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT/Kernel*, 2003.

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

[13] T. Horvath, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *KDD*, 2004.

[14] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *ICDM*, pages 549–552, 2003.

[15] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *NIPS*. 2011.

[16] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, pages 13–23, 2000.

[17] N. Jin, C. Young, and W. Wang. GAIA: graph classification using evolutionary computation. In *SIGMOD*, pages 879–890, 2010.

[18] X. Kong and P. S. Yu. Semi-supervised feature selection for graph classification. In *KDD*, pages 793–802, 2010.

[19] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[20] H. T. Nguyen and A. W. M. . Smeulders. Active learning using pre-cluster. In *ICML*, pages 623–630, 2004.

[21] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *KDD*, pages 647–652, 2004.

[22] M. Opper, H. S. Seung, and H. Sompolinsky. Query by committee. In *COLT*, pages 287–294, 1992.

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

[24] 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 *SDM*, pages 1075–1086, 2009.

[25] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *ICML*, pages 999–1006, 2000.

[26] X. Yan, H. Cheng, J. Han, and P. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, pages 433–444, 2008.

[27] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.

[28] 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.

[29] B. Yang, J. Sun, T. Wang, and Z. Chen. Effective multi-label active learning for text classification. In *KDD*, pages 917–926, 2009.

[30] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *ICML*, pages 1081–1088, 2006.