Graph Classification: A Diversified Discriminative Feature Selection Approach

Yuanyuan Zhu, Jeffrey Xu Yu, Hong Cheng, Lu Qin
The Chinese University of Hong Kong, Hong Kong, China
{yyzhu,yu,hcheng,lqin}@se.cuhk.edu.hk

ABSTRACT

A graph models complex structural relationships among objects, and has been prevalently used in a wide range of applications. Building an automated graph classification model becomes very important for predicting unknown graphs or understanding complex structures between different classes. The graph classification framework being widely used consists of two steps, namely, feature selection and classification. The key issue is how to select important subgraph features from a graph database with a large number of graphs including positive graphs and negative graphs. Given the features selected, a generic classification approach can be used to build a classification model. In this paper, we focus on feature selection. We identify two main issues with the most widely used feature selection approach which is based on a discriminative score to select frequent subgraph features, and introduce a new diversified discriminative score to select features that have a higher diversity. We analyze the properties of the newly proposed diversified discriminative score, and conducted extensive performance studies to demonstrate that such a diversified discriminative score makes positive/negative graphs separable and leads to a higher classification accuracy.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology —*Feature evaluation and selection*

Keywords

Graph Classification, Feature Selection, Diversity

1. INTRODUCTION

A graph models complex structural relationships among objects, and has been prevalently used in a wide range of applications, such as chemical compound structures in chemistry, attributed graphs in image processing, food chains in ecology, electrical circuits in electricity, protein interaction networks in biology, etc. With the increasing popularity of graph databases that contain a large number of graphs in various applications, building an automated classification model emerges as one important problem for predicting the unknown graphs or understanding complex structures between different classes. For example, with a chemical compound dataset, chemists want to be able to predict which chemical compounds are active and which are inactive.

The graph classification framework being widely used is first to select a set of features by mining frequent subgraphs from the graph database that consists of positive graphs and negative graphs, and then to employ a generic classification model using the set of features selected. Such an approach has been shown achieving promising classification accuracy [5], and the key issue is how to select features. In order to achieve a higher classification accuracy, discriminative based approaches have been extensively studied that select a set of discriminative frequent features [20, 16]. The discriminative frequent features mined are the features that are frequent in the positive graph set but are infrequent in the negative graph set.

LEAP [20] is one of the first work that studies how to directly mine discriminative subgraph features instead of finding the discriminative features from the frequent features mined. Some recent work study how to speed up the process of mining discriminative subgraph features such as graphSig [16], COM [11], and GAIA [12]. graphSig [16] provides an efficient solution for mining discriminative subgraph features with extremely low frequencies. First, graphSig converts graphs into feature vectors by performing a random walk with restarts on every vertex. Second, graphSig divides graphs into small groups such that graphs in the same group have similar vectors. Third, graphSig mines frequent subgraphs in each group with high frequency thresholds, because high similarity among vectors in the same group indicates that the corresponding graphs in the group share highly frequent subgraphs. COM [11] takes into account the co-occurrences of subgraph features. Co-occurrences of small subgraph features are able to approximate large features and thereby significantly reduce the mining time for large features. Both graphSig and COM are much faster than LEAP. For accuracy in classifying chemical compounds, COM has a comparable classification accuracy to that of LEAP, and GraphSig produces a higher accuracy than LEAP. GAIA [12] proposes an evolutionary computation method to mine discriminative subgraph features for graph classification using a randomized searching strategy. The randomized search strategy simulates biological evolution to select discriminative subgraph features. GAIA is the up-to-date approach in graph classification.

All the existing works select discriminative subgraph features by a discriminative score such as G-test [20], Fisher score [4] and log ratio score [11, 12]. We classify them as a single feature discriminative score, because they assign a score to a feature by only considering its own occurrences in the positive/negative sets. Among them, the log ratio score shows better classification accuracy as reported in [11, 12].

However, the question to be asked is whether there exists a better score with which a better set of features can be selected and a graph classification model can be built with an even higher accuracy. The reason to ask such a question is that there does not exist a ground truth. First, for graph classification over a large graph database with positive/negative graphs, the number of frequent subgraph features is huge. It is known to be impractical to explore all the possible subsets of frequent subgraph features for classification. Second, it is still an open problem to identify the true optimal feature selection, because the connection between the features selected and the accuracy of the graph classification is unknown yet. Therefore, discriminative score is important, but it may not be sufficient to select features that make the positive and negative graphs separable. We identify two issues with the single feature discriminative score. (Issue-1) The discriminative features selected can be highly overlapped. They may not lead to a good classification accuracy, because one of them is redundant in a sense that they appear frequently together and share a large part between them. (Issue-2) Some important features may be missed out, because there exists a restriction on the number of discriminative features to be selected and selecting highly overlapped discriminative features may squeeze out important features which are less discriminative and barely overlapped with other features. Although feature selection is a well studied problem in the literature for conventional classification tasks on high-dimensional data [4, 15, 6], such methods have limited power for graph datasets. The underlying reasons are: each feature is treated equivalently, and the relevance and redundancy is evaluated by their distributions over binary vectors. However, in a graph database, features have more complicated relationship with each other in terms of the topological structures and the locations. In addition, a pattern has an exponential number of subgraphs, which makes the patterns not uniformly distributed. This leads to a huge redundancy and complex overlapping of patterns.

To address these two issues, in this paper, we study a new score, called a diversified discriminative score. The main idea is to explore the additional value of the diversity together with the discriminativity. By diversity, we explore how to reduce the overlapping between two features that share a large part, based on an edge-cover. In addition, to further enhance diversity, we explore how to reduce the overlapping between a feature and a set of features. We do this because a feature f to be selected may not have a large overlapping with any specific feature already selected, but it can be overlapped with the set of features already selected, i.e., many features in the set are partially overlapped with feature f, and the feature set end up to be overlapped with the major part of feature f. In other words, such a feature f may not add additional value to make positive/negative graphs separable. The main contributions of this paper are summarized below. In addition to the new diversified discriminative score, we also analyze the properties of our newly proposed diversified discriminative score from many viewpoints. We propose algorithms to select such features, and conduct extensive experimental studies. We show that the features selected by the diversified discriminative score outperform the up-to-date graph classification approach GAIA [12].

The rest of the paper is organized as follows. Section 2 gives the problem statement. In Section 3, we discuss two main issues with the existing discriminative scores, and propose a new diversified and discriminative score. In Section 4, we discuss the discriminative power of our new diversified and discriminative score by investigating the statistic information of the new score in comparison with the existing discriminative scores. We give our algorithms in Section 5. We further develop an ensemble method to select features and build classification model in Section 6. Our experimental results are shown in Section 7. We discuss the related work in Section 8, and conclude this paper in Section 9.

$\mathcal{D}$	f_1	f_2	f_3	f_4	f_5	Class Label
P_1	1	0	0	1	0	+
P_2	1	1	1	0	1	+
P_3	1	1	1	0	1	+
P_4	0	1	0	1	1	+
P_5	0	0	1	1	1	+
P_6	0	0	1	1	1	+
N_1	1	0	0	1	0	-
N_2	0	1	0	1	0	-
N_3	0	0	0	0	0	-
N_4	0	0	0	0	1	-
N_5	0	0	1	0	1	-
N_6	0	0	1	0	1	-

Table 1: The Feature Containment

2. PROBLEM STATEMENT

Given a set of labels Σ, we model a graph as $g = (V, E, l)$ where V is the set of vertices, $E \subseteq V \times V$ is the set of edges, and l is a labeling function on vertices and edges, where the label of a vertex $u \in V$ is specified by $l(u)$ and the label of an edge (u, v) is specified by $l(u, v)$. We use $V(g)$ and $E(g)$ to denote the set of vertices and the set of edges of a graph g, respectively, and use $|V(g)|$ and $|E(g)|$ to denote the numbers of vertices and edges in g.

Definition 2.1: Subgraph Isomorphism. Given two graphs $g' = (V', E', l')$ and $g = (V, E, l)$, g' is *subgraph isomorphic* to g, if there exists an injective function $\varphi : V' \to V$ such that (1) for $\forall u, v \in V'$ and $u \neq v$, $\varphi(u) \neq \varphi(v)$, (2) for $\forall v \in V'$, $\varphi(v) \in V$ and $l'(v) = l(\varphi(v))$, and (3) for $\forall (u, v) \in E'$, $(\varphi(u), \varphi(v)) \in E$ and $l(u, v) = l'(\varphi(u), \varphi(v))$. $\square$

A graph g' is a subgraph of another graph g, denoted as $g' \subseteq g$, if g' is *subgraph isomorphic* to g. Here, g is a supergraph of g', denoted as $g \supseteq g'$, if g' is a subgraph of g.

Problem Statement: Let $\mathcal{D} = \{g_1, g_2, \cdots, g_n\}$ be a graph database of size $|\mathcal{D}|$ that consists of many graphs g_i, for $1 \leq i \leq |\mathcal{D}|$. Among the graphs in $\mathcal{D}$, there are a set of positive graphs, denoted as $\mathcal{D}^+$ ($\subseteq \mathcal{D}$), and a set of negative graphs, denoted as $\mathcal{D}^-$ ($\subseteq \mathcal{D}$), where $\mathcal{D}^+ \cap \mathcal{D}^- = \emptyset$. The feature selection problem is to select a set of features $\mathcal{F}_s$ with which a model $\mathcal{M}$ can be built for classifying unseen graphs with a high classification accuracy.

Following the existing works, we select frequent subgraphs (pattern) from $\mathcal{D}$ as features. Given $\mathcal{D}$, the frequency of a feature f is given as

$$freq(f) = \frac{|sup(f, \mathcal{D})|}{|\mathcal{D}|}$$

where $sup(f, \mathcal{D}) = \{g_i \mid g_i \in \mathcal{D}, f \subseteq g_i\}$ is the support set of f. The support of the subgraph f in $\mathcal{D}$ is $|sup(f, \mathcal{D})|$. A subgraph f is called a frequent subgraph if $freq(f) \geq \tau$, where τ ($0 \leq \tau \leq 1$) is a user specified minimum support threshold. For simplicity, we use $sup(f)$ as an alternative notation for $sup(f, \mathcal{D})$ in the following discussions.

Example 2.1: Fig. 1 shows a graph database $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. Here, $\mathcal{D}^+ = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ is a set of positive graphs, and $\mathcal{D}^- = \{N_1, N_2, N_3, N_4, N_5, N_6\}$ is a set of negative graphs. From this graph database, a frequent subgraph set can be mined which might contain a large number of subgraphs. Suppose we select a subset $\mathcal{F}_s = \{f_1, f_2, f_3, f_4, f_5\}$ as the features as shown

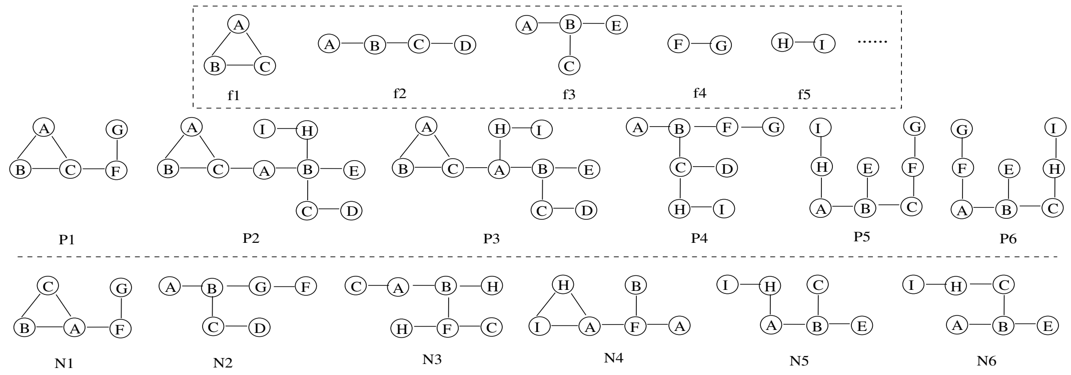


Figure 1: A graph database $\mathcal{D}$ with 6 positive graphs P_i $(1 \leq i \leq 6)$ and 6 negative graphs N_j $(1 \leq j \leq 6)$, and pattern set $\mathcal{F}_s$

.

in Fig. 1. Table 1 shows whether a feature $f \in \mathcal{F}_s$ is a subgraph of P_i $(1 \leq i \leq 6)$ or N_j $(1 \leq j \leq 6)$. The corresponding value is 1 if it is, otherwise 0. Then a generic classification model can be built for the graph database based on Table 1. If we choose another different subset $\mathcal{F}'_s$ as the feature set, the graph database will be mapped to a different vector space which might lead to building a quite different classifier. In other words, the quality of the classifier in terms of the accuracy for a graph database heavily depends on the feature set $\mathcal{F}_s$. □

3. DISCRIMINATIVE SCORE

Frequent features mined from the graph database $\mathcal{D}$ are widely used in graph classification. As observed in many existing works, it does not necessarily mean that all frequent features are equally important for building a good classification model, because some of them may occur frequently in both $\mathcal{D}^+$ and $\mathcal{D}^-$. Such frequent features are not discriminative and consequently do not lead to a good classification model. To address this issue, people explore to use the discriminative frequent features that appear in $\mathcal{D}^+$ frequently but appear in $\mathcal{D}^-$ infrequently. Several statistic measures are proposed to evaluate the discriminative power of a feature, such as G-test [20], Fisher score [4] and log ratio score [11, 12]. We call them as a single feature discriminative score, because they assign a score to a feature by merely considering its own occurrences in the positive/negative graph sets.

3.1 The Single Feature Discriminative Score

In this work, we focus on the log ratio score which has recently been widely used due to the better classification accuracy as reported in [11, 12].

Given a graph database $\mathcal{D}$ with a set of positive graphs, $\mathcal{D}^+$, and a set of negative graphs, $\mathcal{D}^-$. The positive and negative frequencies of a subgraph feature f, denoted as $r^+(f)$ and $r^-(f)$, are defined as follows.

$$\begin{cases} r^+(f) = \dfrac{|sup(f, \mathcal{D}^+)|}{|\mathcal{D}^+|} \\ r^-(f) = \dfrac{|sup(f, \mathcal{D}^-)|}{|\mathcal{D}^-|} \end{cases}$$

The discriminative score of feature f by the log ratio is defined below.

$$score(f) = \log \frac{r^+(f)}{r^-(f)} \tag{1}$$

It considers the log ratio of the positive and negative frequencies. To avoid using a zero denominator, when calculating the negative frequencies, it adds a virtual negative graph that contains any subgraph features.

Given the log ratio as a discriminative score, a commonly used approach is to select the feature that has the largest discriminative score from the unselected features iteratively. For example, the works in [11, 12] consider to extend a subgraph to generate its supergraphs only based on its discriminative score or the score increment. However, there are two main issues with the existing feature selection approaches.

- **Issue-1: Highly Overlapped Features**. The discriminative features selected can be possibly highly overlapped. In other words, two discriminative features, f_i and f_j, may share a large common part. They may not lead to a good classification accuracy, because one of them is redundant in a sense that they appear frequently together in graphs in $\mathcal{D}$, as guided by the definition of log ratio.
- **Issue-2: Missing Important Features**. Some important features may be missed out. The number of discriminative features selected for building a classifier is often restricted due to efficiency and effectiveness considerations. As discussed in Issue-1, the highly overlapped discriminative features may squeeze out the important features which are less discriminative and barely overlapped with other features due to the limited feature number.

3.2 A New Diversified Discriminative Score

Considering these two issues, in this section, we give a new diversified discriminative score. First, we introduce the concept of embedding.

Definition 3.1: Embedding. Given two graphs f and g. Suppose $f \subseteq g$ by a subgraph isomorphic injection function φ. The subset of vertices of $V(g)$, $V(g)_\varphi = \{\varphi(u) | u \in V(f)\}$, is a vertex embedding of f in g. In a similar way, the subset of edges of $E(g)$, $E(g)_\varphi = \{(\varphi(v), \varphi(u)) | (v, u) \in E(f)\}$ is an edge embedding of f in g. □

With edge embedding, we consider a feature f from the viewpoint of coverage. A feature f covers certain edges in g when there is an edge embedding of f in g. With such coverage, we further explore how many edges a feature f can cover in graph database $\mathcal{D}$, which is the union of edges covered by f in all graphs in $\mathcal{D}$.

Definition 3.2: Feature Edge-Cover. Given a graph database $\mathcal{D} = \{g_1, g_2, \cdots, g_n\}$ and a feature f, a graph $g_i \in \mathcal{D}$ is covered by f if there is an edge embedding of f in g_i, and $e \in E(g_i)$ is covered by f if $e \in E(g_i)_\varphi$. We denote the set of edges in g covered by feature f as $E_c(f, g)$. The feature edge-cover of f is the set of edges that are covered by f in all graphs in $\mathcal{D}$, denoted as $C_e(f) = \cup_{g_i \in sup(f)} E_c(f, g_i)$. The edge-cover score of f is the number of edges covered by f, denoted as $S_c(f) = \sum_{g_i \in sup(f)} |E_c(f, g_i)|$. We use $S_c(f)$ and $|C_e(f)|$ alternatively in the following discussion since $S_c(f) = |C_e(f)|$. □

A main idea behind the feature edge-cover of f is that it considers the embedding locations in every graph g_i in $\mathcal{D}$. Such implicit locations assist us to select diversified features in a sense that they are not overlapped with the features selected already, as we will discuss in the following. For the purpose of addressing the diversity as an answer to the two issues, we introduce an edge-cover probability. The edge-cover probability of a feature f regarding to the entire edge set in $\mathcal{D}$ is defined in Eq. (2).

$$p_c(f) = \frac{|C_e(f)|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \tag{2}$$

Consider the diversity from the viewpoint of independence. If the graph features are independent of each other, the conditional probability of a feature with respect to another graph feature will be the same prior edge-cover probability of itself. In other words, given two features f_1 and f_2, they are independent if $p_c(f_2|f_1) = p_c(f_2)$. It indicates that two features can not be considered as diversified in terms of independence, if their edge-covers are largely overlapped. Therefore, the decision for selecting a feature f_2 with a feature f_1 already selected depends on the probability $p_c(f_2|f_1)$. Based on this consideration, we define a conditional edge-cover probability of feature f_2 with respect to another feature f_1 as follows.

$$p_c(f_2|f_1) = \frac{|C_e(f_1 \cap f_2)|}{|C_e(f_1)|} \tag{3}$$

where $C_e(f_1 \cap f_2) = C_e(f_1) \cap C_e(f_1)$. The joint edge-cover probability of f_1 and f_2 can be derived in the following.

$$\begin{aligned} p_c(f_1 \cap f_2) &= p_c(f_2|f_1) \cdot p_c(f_1) \\ &= \frac{|C_e(f_1 \cap f_2)|}{|C_e(f_1)|} \cdot \frac{|C_e(f_1)|}{\sum_{g_i \in D} |E(g_i)|} \\ &= \frac{|C_e(f_1 \cap f_2)|}{\sum_{g_i \in D} |E(g_i)|} \end{aligned} \tag{4}$$

Note that the value of $p_c(f_1 \cap f_2)$ is bounded by [0,1] and is also a symmetric measure, which means that $p_c(f_1 \cap f_2) = p_c(f_1|f_2) \cdot p_c(f_2)$ also holds. Based on Eq. (4), the union edge-cover probability of feature f_1 and f_2 can be derived as follows.

$$p_c(f_1 \cup f_2) = p_c(f_1) + p_c(f_2) - p_c(f_1 \cap f_2) \tag{5}$$

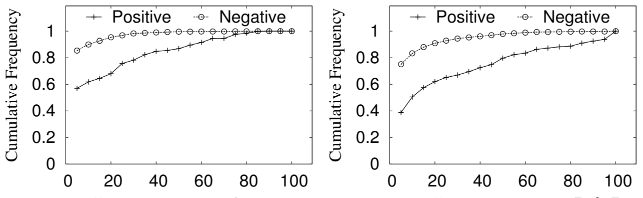
We extend the union edge-cover probability from two features to m features where $m > 2$. Let $\mathcal{F} = \{f_1, \cdots, f_m\}$ be a set of features, and let $\mathcal{F}_i = \{f_1, \cdots, f_i\}$ be a subset of $\mathcal{F}$ for $i = 1, \ldots, m$. The edge-cover probability of feature set $\mathcal{F}_m$ can be derived as follows.

$$\begin{aligned} p_c(\mathcal{F}_m) &= p_c(\bigcup_{i=1}^{m} f_i) \\ &= p_c(f_m) + p_c(\mathcal{F}_{m-1}) - p_c(f_m \cap \mathcal{F}_{m-1}) \end{aligned} \tag{6}$$

Thus, the diversity of f_m to be selected with respect to a set of features selected, $\mathcal{F}_{m-1}$, can be derived below.

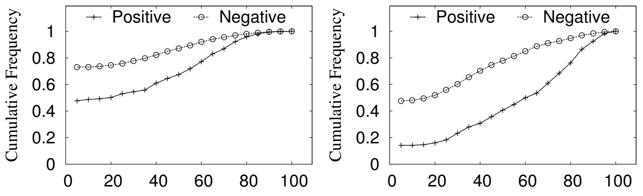
$$\begin{aligned} p_d(f_m) &= p_c(\mathcal{F}_m) - p_c(\mathcal{F}_{m-1}) = p_c(f_m) - p_c(f_m \cap \mathcal{F}_{m-1}) \\ &= \frac{|C_e(f_m) - C_e(f_m \cap \mathcal{F}_{m-1})|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \\ &= \frac{|C_e(f_m) - \bigcup_{j=1}^{m-1} C_e(f_m \cap f_j)|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \\ &= \frac{\sum_{g_i \in sup(f_m)} |E_c(f_m, g_i)|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \\ &\quad - \frac{\sum_{g_i \in sup(f_m)} |\bigcup_{j=1}^{m-1} (E_c(f_m, g_i) \cap E_c(f_j, g_i))|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \end{aligned} \tag{7}$$

Equipped with the diversity of f_m, we define a new diversified discriminative score for a new feature, f_m, to be selected with a feature set $\mathcal{F}_{m-1}$ that has been selected already. Here, let $p_d^+(f_m)$ be the diversity of f_m for the set of positive graphs ($\mathcal{D}^+$), and let $p_d^-(f_m)$ be the diversity of f_m for the set of negative graphs ($\mathcal{D}^-$). The diversified discriminative score of feature f_m is defined as follows.



(a) # of Features (Logratio) (b) # of Features (D&D)

Figure 2: Cumulative Frequency of Features



(a) # of Embeddings (Logratio) (b) # of Embeddings (D&D)

Figure 3: Cumulative Frequency of Embeddings

$$score_d(f_m) = \log \frac{p_d^+(f_m)}{p_d^-(f_m)} \tag{8}$$

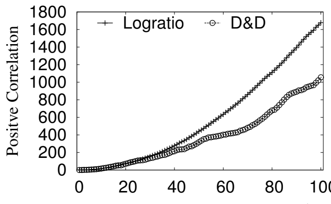
Our proposed diversified discriminative score measures the feature overlapping by evaluating their embeddings in graphs. If most of the embeddings of a new feature f_m have already been covered by the feature subset $\mathcal{F}_{m-1}$, the diversified discriminative score of f_m will be very low, thus f_m will not be selected, even though the log ratio score of f_m can be large. The diversified discriminative score measure can better address the two issues, namely, Issue-1 and Issue-2.

4. PROPERTY STATISTICS OF DISCRIMINATIVE SCORE

In this section, we discuss the discriminative power of our new diversified discriminative score by investigating the statistic information of our new score in comparison with the single feature discriminative score, log ratio, using a real chemical compound dataset containing 400 positive graphs and 1600 negative graphs. We compare two feature sets, each containing 100 features selected by two methods: Logratio and D&D. Logratio is a method that selects frequent features with the maximum single feature discriminative score defined by log ratio (Eq. (1)) iteratively. D&D is our proposed method that selects frequent features using the diversified discriminative score (Eq. (8)). We reemphasize the point that the diversified discriminative score controls the feature diversity by reducing the feature overlapping between features. We will discuss D&D in detail in Section 5. Below, we show how the two feature sets exhibit different properties in separating positive graphs from negative graphs.

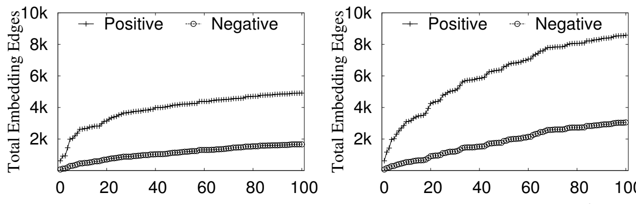
Cumulative Frequencies: For each feature set selected by either Logratio or D&D, we count the number of features contained by each graph in the set of positive/negative graphs, and compute a cumulative frequency. The cumulative frequency for a number i is the percentage of positive (or negative) graphs that have no more than i features. We plot the cumulative frequencies in Fig. 2(a) and Fig. 2(b), for i to be taken from 1 to 100.

For Logratio, as shown in Fig. 2(a), we can see that the positive cumulative frequencies and the negative cumulative frequencies,



(a) # of Features in $\mathcal{D}^+$ (b) # of Features in $\mathcal{D}^-$

Figure 4: Correlation of Features



(a) # of Features (Logratio) (b) # of Features (D&D)

Figure 5: Total Embedding Edges Statistic

based on the positive graphs and the negative graphs respectively, are very close. This means that the feature occurrences in the positive and negative graphs are very similar, thus the features selected by the log ratio can not help too much to distinguish the positive graphs from the negative graphs. Therefore, there is a limit to build a classification model with high accuracy using the single feature discriminative score. On the other hand, for D&D, as shown in Fig. 2(b), the gap between the positive cumulative frequencies and the negative cumulative frequencies is larger. This means that the selected feature set makes the positives and negatives more separable. In addition, Fig. 3(a) and Fig. 3(b) show the cumulative frequencies of positive graphs and negative graphs containing the feature embeddings from the two selected feature sets. We can observe similar trends on the two feature sets. And the gap between the cumulative frequencies of positive graphs ($\mathcal{D}^+$) and negative graphs ($\mathcal{D}^-$) becomes larger with the new diversified discriminative score as shown in Fig. 3(b).

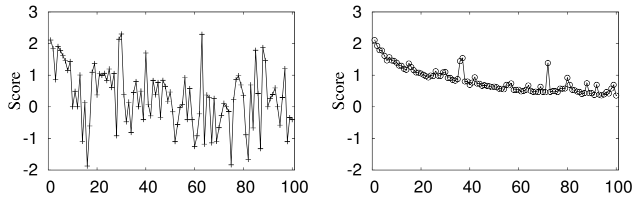
Correlation: The correlation between two features is an important indicator which can be evaluated by comparing their support sets [20]. A widely used measure is Jaccard Coefficient [4], which is defined as follows.

$$corr(f_1, f_2) = \frac{|sup(f_1) \cap sup(f_2)|}{|sup(f_1)| + |sup(f_2)| - |sup(f_1) \cap sup(f_2)|}$$

where $corr(f_1, f_2) \in [0, 1]$. Over a set of 100 features, the sum of pairwise correlations score is a number in the range of [0,10000]. Fig. 4 shows the sum of correlation scores between features when the number of features increases. Our new diversified discriminative score by D&D can reduce about 1/3 correlation score than that by Logratio, which means the features selected by Logratio are more redundant.

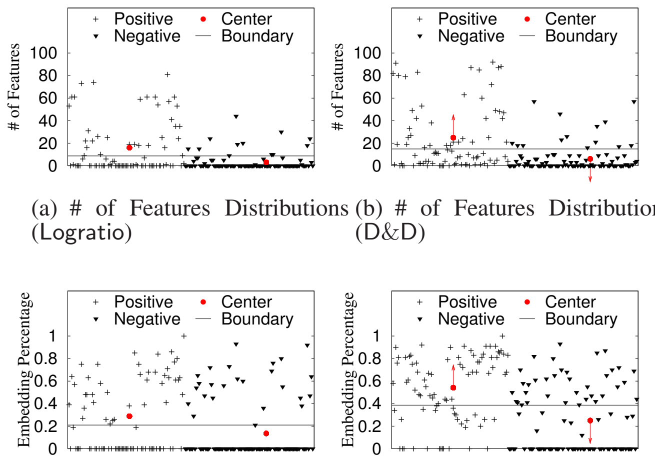
Total Embedding Edges: We count the total embedding edges for a feature set, which is the union of the feature edge-covers of all features in the feature set. Fig. 5(a) and Fig. 5(b) show the total embedding edges by the two feature sets selected by Logratio and D&D respectively. We can see that the gap between positive graphs and negative graphs with D&D in Fig. 5(b) is larger than that with Logratio in Fig. 5(a). Thus, the feature set selected by D&D can cover many edges in positive graphs and few edges in negative graphs and make positive and negative graphs more separable.

Diversified Discriminative Power: Fig. 6 shows the diversified discriminative score of the i_{th} feature being selected. Fig. 6(b) shows that D&D has steadily high diversified discriminative scores for the selected features. We also show the diversified discriminative scores for the features selected by Logratio in Fig. 6(a). As can be seen from Fig. 6(a), it has low and unstable diversified discriminative scores for the selected features, due to the large overlap between the selected features.



(a) i_{th} Feature (Logratio) (b) i_{th} Feature (D&D)

Figure 6: Feature Diversified Discriminative Score



(a) # of Features Distributions (Logratio) (b) # of Features Distributions (D&D)

(c) Embedding Percentage Distributions (Logratio) (d) Embedding Percentage Distributions (D&D)

Figure 7: Statistic of Graphs

The Distributions: With the 100 features selected by Logratio or D&D, we show the feature distribution and embedding distribution on a graph dataset with 200 representative graphs (100 positive graphs and 100 negative graphs) among 2,000 graphs being tested. Fig. 7(a) and Fig. 7(b) show the number of features contained by each positive graph and negative graph, given the 100 feature sets selected by Logratio and D&D, respectively. The boundary line represents the average number of features contained by the 200 sample graphs. The two center points represent the average number of features in positive graph set and negative graph set respectively, where the positive center is above the boundary and the negative center is below the boundary. The positive and negative graphs in Fig. 7(b) spread farther apart based on the feature containment information, which indicates that our new diversified discriminative score selects a feature set which makes positive and negative graphs more separable. Similarly, Fig. 7(c) and Fig. 7(d) show that the embedding percentage in each positive graph and negative graphs. The embedding percentage for a graph is the ratio of the number of embedding edges in the graph to the number of graph edges. These two figures have a similar trend as shown in Fig. 7(a) and Fig. 7(b).

Based on the statistic information given above, it is clear that our proposed diversified discriminative score can help select more discriminative features with little overlap, thus make the positive and negative graphs more separable. With such a score, we are in a much better position to address the two issues, namely, Issue-1 and Issue-2.

Algorithm 1 D&D $(\mathcal{D}, \mathcal{F}, k)$

```
Input: A graph database $\mathcal{D} = (\mathcal{D}^+, \mathcal{D}^-)$, a set of frequent features $\mathcal{F}$,
    and the number of features $k$
Output: A selected feature set $\mathcal{F}_s$
1: $f \leftarrow \text{argmax}_{f \in \mathcal{F}} score(f)$;
2: $\mathcal{F}_s \leftarrow \{f\}$;
3: while $|\mathcal{F}_s| \leq k$ do
4:     $f \leftarrow \text{argmax}_{f \in \mathcal{F} \setminus \mathcal{F}_s} score_d(f)$;
5:     $\mathcal{F}_s \leftarrow \mathcal{F}_s \cup \{f\}$;
6: return $\mathcal{F}_s$;
```

Algorithm 2 D&D-Fast$(\mathcal{D}, \mathcal{F}, k)$

```
Input: A graph database $\mathcal{D} = (\mathcal{D}^+, \mathcal{D}^-)$, a set of frequent features $\mathcal{F}$,
    and the number of features $k$
Output: A selected feature set $\mathcal{F}_s$
1: $\mathcal{F}_s \leftarrow \emptyset$;
2: $\mathcal{H} \leftarrow$ max-heap initialized to be $\emptyset$;
3: for all $f \in \mathcal{F}$ do
4:     $\mathcal{H}.push(f, score(f))$;
5: while $|\mathcal{F}_s| \leq k$ do
6:     $h \leftarrow \mathcal{H}.pop()$;
7:     $\mathcal{F}_s \leftarrow \mathcal{F}_s \cup \{f'\}$; {$f'$ is obtained from $h$.}
8:     update $(\mathcal{H}, \mathcal{F}, \mathcal{F}_s, f')$;
9: return $\mathcal{F}_s$;

10: Procedure update $(\mathcal{H}, \mathcal{F}, \mathcal{F}_s, f')$
11: for all $f \in \mathcal{F} \setminus \mathcal{F}_s$ do
12:     $C_e^-(f) \leftarrow C_e^-(f) - C_e^-(f) \cap C_e^-(f')$;
13:     compute $\overline{score}_d(f)$;
14:     if $\overline{score}_d(f) > head(\mathcal{H}).score_d()$ then
15:         $C_e^+(f) \leftarrow C_e^+(f) - C_e^+(f) \cap C_e^+(f')$;
16:         compute $score_d(f)$;
17:         $\mathcal{H}.update((f, score_d(f)))$;
18:     else
19:         $\mathcal{H}.update((f, \overline{score}_d(f)))$;
```

5. THE ALGORITHMS

In this section, we show the algorithm, named D&D, to select features based on the diversified discriminative score. The D&D algorithm is given in Algorithm 1. There are three inputs: a graph database $\mathcal{D}$ which contains a set of positive graphs $\mathcal{D}^+$ and a set of negative graphs $\mathcal{D}^-$, a set of frequent features (subgraphs) mined from the graph database $\mathcal{D}$ using [21], and a number k which indicates how many diversified discriminative features to be selected from $\mathcal{F}$. In the D&D algorithm, first, it selects the feature f with the largest log ratio score using $score(f)$ (Eq. (1)). This feature f has the highest discriminative power by itself. Then, in a while loop, it iteratively picks up a new feature f with the largest diversified discriminative score $score_d(f)$ (Eq. (8)). This process repeats until k features are selected.

To compute $score_d(f)$, it needs to compute $p_d(f)$ (Eq. (7)) first which involves a feature edge-cover intersection operation, as we describe below. This operation takes two feature edge-covers, $C_e(f)$ and $C_e(f')$, as input and produces their intersections $C_e(f \cap f')$. In this operation, we need first find the graphs that contained by both $sup(f)$ and $sup(f')$ and then compute the intersection of edge embeddings of f and f' for each of these graphs. Thus, the complexity of feature edge-cover intersection operation is $O(|s|_{max} \cdot |f|_{max})$, where $|s|_{max}$ is the maximum support size and $|f|_{max}$ is the maximum feature size.

In Algorithm 1, we need to compute $score_d(f)$ for each feature $f \in \mathcal{F} \setminus \mathcal{F}_s$ to select a feature with the maximum score value in line 4. To compute $score_d(f)$, we need to compute the intersection of $C_e(f)$ and $C_e(f')$ for each $f' \in \mathcal{F}_s$ and the union them. Hence, to select a feature with the maximum diversified discriminative score from $\mathcal{F} \setminus \mathcal{F}_s$, it needs $|\mathcal{F}_s| \cdot |\mathcal{F} \setminus \mathcal{F}_s|$ feature edge-cover intersection operations. To find the set of k features, the whole algorithm ends up with $O(k^2 \cdot |\mathcal{F}|)$ feature edge-covers operations, where k is the number of features to be selected.

To reduce the time complexity, we design a faster algorithm which only needs $O(k \cdot |\mathcal{F}|)$ feature edge-cover intersection operations. The algorithm, named D&D-Fast, is shown in Algorithm 2. For each candidate feature f_j, we can record its two edge-cover scores. One is the positive edge-cover score, $|C_e^+(f_j)|$, based on the positive graph set $\mathcal{D}^+$, and the other is the negative edge-cover score, $|C_e^-(f_j)|$, based on the negative graph set $\mathcal{D}^-$. We update these scores in every iteration. Let these scores for a feature f_j at the i_{th} iteration be $|C_e^+(f_j)_i|$ and $|C_e^-(f_j)_i|$. Suppose at the i_{th} iteration, a feature f' with the maximum diversified discriminative score is selected. Then for each feature $f_j \in \mathcal{F} \setminus (\mathcal{F}_s \cup \{f'\})$, we update $C_e^+(f_j)_i$ and $C_e^-(f_j)_i$ as follows.

$$\begin{aligned} C_e^+(f_j)_i &= C_e^+(f_j)_{i-1} - C_e^+(f_j)_{i-1} \cap C_e^+(f') \\ C_e^-(f_j)_i &= C_e^-(f_j)_{i-1} - C_e^-(f_j)_{i-1} \cap C_e^-(f') \end{aligned} \tag{9}$$

By updating the edge-cover score for each feature $f_j \in \mathcal{F} \setminus (\mathcal{F}_s \cup \{f'\})$ immediately after the selection of a new feature f', we avoid computing the intersection of $C_e(f)$ and $C_e(f')$ for each $f' \in \mathcal{F}_s$ from scratch, and only need $O(k \cdot |\mathcal{F}|)$ feature edge-cover intersection operations.

In addition, we can further speed up the feature edge-cover intersection operation by an early stop bound of $score_d(f_j)$. We use a max-heap $\mathcal{H}$ to maintain the current $score_d(f_j)$ for each $f_j \in \mathcal{F} \setminus \mathcal{F}_s$ which is sorted according to the value of $score_d(f_j)$. Each entry in the heap is with the form $e = (pID, score_d(f_j))$, where pID is the feature ID and $score_d(f_j)$ is the score to be sorted in the heap. Note that we do not need to update $score_d(f_j)$ for all $f_j \in \mathcal{F} \setminus \mathcal{F}_s$. We can derive an upper bound of $score_d(f_j)$ as follows.

$$\begin{aligned} score_d(f_j)_i &= \log \frac{p_d^+(f_j)_i}{p_d^-(f_j)_i} \\ &= \log \frac{|C_e^+(f_j)_{i-1} - C_e^+(f_j)_{i-1} \cap C_e^+(f')|}{|C_e^-(f_j)_{i-1} - C_e^-(f_j)_{i-1} \cap C_e^-(f')|} \\ &\leq \log \frac{|C_e^+(f_j)_{i-1}|}{|C_e^-(f_j)_{i-1} - C_e^-(f_j)_{i-1} \cap C_e^-(f')|} \\ &= \overline{score}_d(f_j)_i \end{aligned} \tag{10}$$

where $\overline{score}_d(f_j)_i$ is an upper bound of $score_d(f_j)_i$. In Algorithm 2, before updating $C_e^+(f)$ for each f in the i_{th} iteration (line 15), we first update the edge-cover score $C_e^-(f)$ (line 12) and compute the upper bound $\overline{score}_d(f)$ (line 13). If $\overline{score}_d(f)$ is larger than maximum score in the heap, we will updating $C_e^+(f)$ and compute $score_d(f)$. Otherwise, we will skip updating $C_e^+(f)$ and record i. If in the i'_{th} iteration ($i < i' < k$), $\overline{score}_d(f)$ is no longer larger than the maximum score in the heap, we will update $C_e^+(f)$ by considering the i_{th} to i'_{th} features in $\mathcal{F}_s$. By this upper bound, we can skip $C_e^+(f)$ computation for many $f \in \mathcal{F} \setminus \mathcal{F}_s$, which can speed up the feature edge-cover intersection operation. The updated procedure is shown in Algorithm 2 (line 10-19).

6. ENSEMBLE D&D

We discuss a new ensemble approach. In the D&D method, we select a subset of diversified and discriminative features $\mathcal{F}_s$ from a given feature set. The selected features will complementarily enhance each other to build a good classifier. Since different feature set $\mathcal{F}_s$ might lead to different classifiers. One feature set might lead to a classifier which can correctly classify part of of training graphs, while a different feature set might lead to another classifier which can correctly classify a different part of training graphs. Hence, after building one classifier for feature subset $\mathcal{F}_s$, we adopt an iterative procedure to focus on graphs that are incorrectly predicted by previous classifiers to improve the performance.

The whole ensemble process of is a boosting like method. We utilize the weight of graph to guide the features selection process. We iteratively build a set of base classifiers from the training data and perform classification by taking a weighted vote on the predictions made by each base classifier. The voting weight of a base classifier C_i depends on its accuracy. As in [11, 12], we use the normalized accuracy which is defined as follows.

$$\begin{aligned} Sensitivity &= \frac{\#\, of\ true\ positives}{\#\, of\ positives} \\ Specificity &= \frac{\#\, of\ true\ negatives}{\#\, of\ negatives} \\ Normalized\ accuracy &= \frac{Sensitivity + Specificity}{2} \end{aligned} \tag{11}$$

The voting weight and the weights of training graphs are defined and updated based on AdaBoost. Suppose the accuracy for classifier C_i is a_i, its voting weight is defined as $b_i = \frac{1}{2}\ln(\frac{a_i}{1-a_i})$. Suppose the weight of each graph is initially set to 1. Then in iteration i, the weight of each graph g_j is updated as follows

$$w_j^{i+1} = \frac{w_j^i}{Z_i} \times \begin{cases} \exp^{-b_i} & \text{if } g_j \text{ is correctly classified} \\ \exp^{b_i} & \text{if } g_j \text{ is misclassified} \end{cases} \tag{12}$$

where Z_i is the normalization factor used to ensure that $\sum_{g_j \in \mathcal{D}} w_j^{i+1} = |\mathcal{D}|$. The weight update formula given in Eq. (12) increases the weights of incorrectly classified graphs and decreases the weights of those classified correctly.

After updating the weights of graphs in the iteration i, we use such weights to guide our feature selection process in the next iteration $i+1$. In doing so, we extend the edge-cover score defined in Definition 3.2 to a weighted edge-cover score as follows.

$$S'_c(f) = \sum_{g_i \in sup(f)} w_i \times |E_c(f, g_i)| \tag{13}$$

Accordingly, the diversity of feature f_m defined in Eq. (7) can be update as follows.

$$\begin{aligned} p'_d(f_m) = &\frac{\sum_{g_i \in sup(f_m)} w_i \times |E_c(f_m, g_i)|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \\ &- \frac{\sum_{g_i \in sup(f_m)} w_i \times |\bigcup_{j=1}^{m-1}(E_c(f_m, g_i) \cap E_c(f_j, g_i))|}{\sum_{g_i \in \mathcal{D}} |E(g_i)|} \end{aligned} \tag{14}$$

Finally , we update the diversified discriminative score in Eq. (8) as follows.

$$score'_d(f_m) = \log \frac{p'^{+}_d(f_m)}{p'^{-}_d(f_m)} \tag{15}$$

Based on Eq. (15), if a pattern f_m covers a lot of misclassified positive graphs, $score'_d(f_m)$ will increase, so f_m will have a higher chance to be selected. If a pattern f_m covers a lot of misclassified negative graphs, $score'_d(f_m)$ will decrease and f_m will have less chance to be selected. By assigning a weight to a graph, we can choose the patterns that are in the misclassified positive graphs, and discard the patterns that frequently occur in the misclassified negative graphs in the next iteration to improve accuracy.

7. PERFORMANCE STUDIES

In this section, we evaluate our algorithms from three aspects. First, we evaluate our D&D and ensemble D&D feature selection criteria. We compare D&D with two feature selection strategies: maximum single feature discriminative score selection and conventional feature selection strategy on high-dimensional data which considers the relevance and redundancy of features. We denote the former as Logratio, and for the latter, we use the latest work MMRFS [4] as a representative of the recent existing works [4, 6, 15]. Second, we compare our method with three alternative graph classification approaches: GAIA [12], COM [11], and graphSig [16]. Third, we test the performance of the algorithms on the pattern set mined by GAIA. We implemented our algorithms, Logratio, and MMRFS using C++ and obtained the GAIA implementation from the authors which also includes the implementation of COM and graphSig. For our algorithm implementations, we use LIBSVM [2] as the classification model. As widely used in previous work [12, 11, 16, 20], the classification accuracy is evaluated with 5-fold cross validation and the normalized accuracy defined in Eq. (11) is computed for each classifier. All tests were conducted on a PC with 2.66GHz CPU and 3.43GB memory running Windows XP.

We evaluate the algorithms on two real datasets: chemical compound datasets and protein datasets, which are also used in [20, 11, 12].

Chemical datasets: Chemical compound datasets are available at PubChem (`http://pubchem.ncbi.nlm.nih.gov`). In PubChem, each dataset belongs to a certain type of cancer screen with the outcome active or inactive. There are 11 graph datasets classified by their biological activities and the detailed description is provided in Table 2. These are all the bioassays used in [20] and [12]. Each compound can be either active or inactive in a bioas-

Table 2: List of Selected Bioassays

ID	Assay ID	Tumor description	Total number of actives	Total number of inactives
1	83	Breast	2287	25510
2	123	Leukemia	3123	36741
3	1	Non-Small Cell Lung	2047	38410
4	109	Ovarian	2072	38551
5	330	Leukemia	2194	38799
6	41	Prostate	1568	25967
7	47	Central Nerv Sys	2018	38350
8	145	Renal	1948	38157
9	81	Colon	2401	38236
10	33	Melanoma	1642	38456
11	167	Yeast anticancer	9467	69998

Table 3: List of Selected SCOP Families

ID	SCOP ID	Family name	Number of selected proteins
1	46463	Globins	51
2	47617	Glutathione S-transferase (GST)	36
3	48623	Vertebrate phospholipase A2	29
4	48942	C1 set domains	38
5	50514	Eukaryotic proteases	44
6	51012	alpha-Amylases, C-terminal beta-sheet domain	26
7	51487	beta-glycanases	32
8	51751	Tyrosine-dependent oxidoreductases	65
9	51800	Glyceraldehyde-3-phosphate dehydrogenase-like	34
10	52541	Nucleotide and nucleoside kinases	27
11	52592	G proteins	33
12	53851	Phosphate binding protein-like	32
13	56251	Proteasome subunits	35
14	56437	C-type lectin domains	38
15	88634	Picornaviridae-like VP	39
16	88854	Protein kinases, catalytic subunit	41

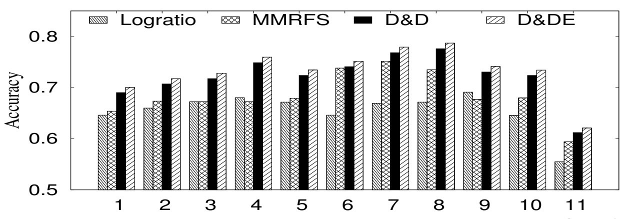


Figure 8: Accuracy Comparison for Logratio, MMRFS, D&D **and** D&DE **(Balanced Chemical Datasets)**

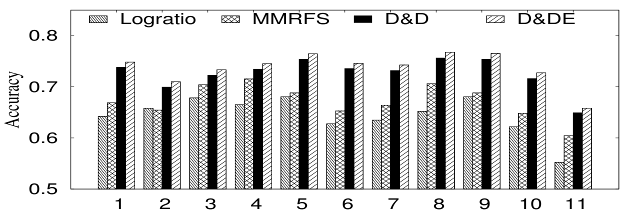


Figure 9: Accuracy Comparison for Logratio, MMRFS, D&D **and** D&DE **(Unbalanced Chemical Datasets)**

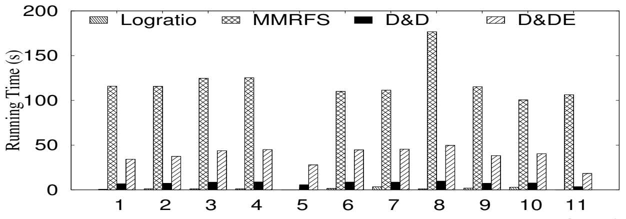


Figure 10: Runtime Comparison for Logratio, MMRFS, D&D **and** D&DE **(Unbalanced Chemical Datasets)**

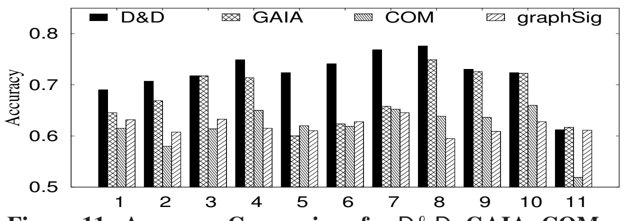


Figure 11: Accuracy Comparison for D&D**, GAIA, COM and graphSig (Balanced Chemical Datasets)**

say. As in [12], we randomly select 400 active compounds as the positive set and 400 inactive compounds as the negative set as a balanced dataset for each bioassay. We also randomly select 400 active compounds and 1,600 negative compounds as an unbalanced dataset. The graph representation of compounds is straightforward. We remove hydrogen atoms, represent each atom by a node labeled with the atom type and each chemical bond by an edge labeled with the bond type. On average, each compound graph has about 25 n-odes and 27 edges.

Protein datasets: The protein datasets consist of protein structures from Protein Data Bank (`http://www.rcsb.org/pdb/`) classified by SCOP (`http://scop.mrc-lmb.cam.ac.uk/scop/`). We use 16 protein datasets generated from all the large SCOP families with more than 25 members (listed in Table 3). In each dataset, protein structures in a selected family are taken as the positive set. Unless otherwise specified, we randomly select 256 other proteins (i.e., not members of the 16 families) as a common negative set used by all protein datasets. For each family, the selected proteins (as the positive graphs) and the 256 negative graphs form a dataset which is unbalanced. We also create a balanced dataset for each family by selecting the same number of negative graphs from the 256 proteins as the number of positive ones in that family. To generate a protein graph, each graph node denotes an amino acid, whose location is represented by the location of its alpha carbon. There is an edge between two nodes if the distance between the two alpha carbons is less than 11.5 angstroms. Nodes are labeled with their amino acid type and edges are labeled with the distances between the alpha carbons. On average, each protein graph has 250 nodes and 2,700 edges.

7.1 D&D Performance Analysis

In this subsection, we study the performance of our method D&D by comparing it with two feature selection strategies: Logratio and MMRFS. Logratio incrementally selects frequent features with maximal single feature discriminative score. MMRFS iteratively selects features with the maximum marginal relevance. To investigate our ensemble strategy, we also compare the performance of a single D&D and ensemble D&D of our method, denoted as D&D and D&DE respectively. Unless other specified, for all algorithms, we select the features from a given feature set mined by gSpan [21] with a minimum support 10%. Since our method is independent of the mining method, we only report the time of feature selection and classification. In this subsection, we only evaluate algorithms on chemical datasets since graphs in protein datasets are very large and gSpan can not mine frequent patterns within reasonable time.

Fig. 8 shows the classification accuracy by different method-s on balanced chemical datasets. Our method D&D outperforms Logratio and MMRFS on the total 11 chemical datasets. Logratio performs worst among all the methods. This shows that the single feature discriminative score in feature selection is not sufficient. MMRFS performs better than Logratio, which proves taking redundancy into consideration in feature selection is necessary. MMRFS is outperformed by D&D due to its limitation on capturing the graph structure. D&DE can get better accuracy than D&D. Fig. 9 shows the classification accuracy by different methods on unbalanced chemical datasets, which has a similar trend as Fig. 8.

The time cost for these algorithms have a similar trend on balanced datasets and unbalanced datasets. Due to the lack of space, we only show the time for unbalanced datasets since they are larg-er than balanced datasets and usually need more time. Fig. 10 shows running time for different methods on unbalanced chemical datasets. Logratio is the fastest because it only needs to sort the single feature discriminative score for each pattern. MMRFS is the most costly since each time it needs to scan all the rest of features to select the one with the maximum marginal score, and in each marginal score computation, it needs to compute the redundancy between the new feature with the already selected $k-1$ features. Our D&D method with the best performance on accuracy is efficient, based on our proposed gradual update strategy and the early stop bound based pruning.

7.2 Comparison with Existing Algorithms

In this section, we compare our method D&D with GAIA, COM and graphSig on chemical datasets. For these three algorithms, we use the parameters that deliver the best results as suggested in [12, 11, 16]. We will report the evaluation results on protein datasets in next subsection.

Fig. 11 shows the classification accuracy by different methods on the 11 balanced chemical datasets. Our method wins on 10 out of the 11 datasets. This clearly demonstrates the effectiveness of our graph features, when the feature overlap is considered based on our proposed diversified discriminative score. With this score measure, more important features with little overlap will be picked.

Fig. 12 shows the classification accuracy by different classification methods on the 11 unbalanced chemical datasets. As in

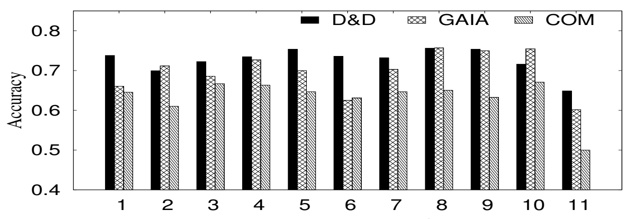


Figure 12: Accuracy Comparison for D&D, GAIA and COM (Unbalanced Chemical Datasets)

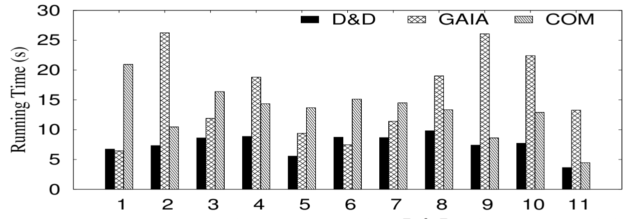


Figure 13: Runtime Comparison for D&D, GAIA and COM (Unbalanced Chemical Datasets)

[12], we did not compare with graphSig since it can not handle unbalanced datasets. Our method wins on 9 out of the 11 datasets and GAIA wins on the other 2. This result again demonstrates the superiority of our diversified discriminative score. Fig. 13 shows running time by different methods on the 11 unbalanced chemical datasets. On most of the datasets, our method is the most efficient, based on our proposed gradual update strategy and the early stop bound based pruning

7.3 Performance on Patterns Mined by GAIA

Our D&D feature selection method is independent of the frequent pattern mining process, because D&D can perform on any given feature set $\mathcal{F}$. We validate the power of D&D feature selection method on a given frequent pattern set mined by gSpan, which usually contains a large number of features. In this subsection, we investigate the performance of D&D on any given incomplete frequent pattern set with a small number of features. We take the pattern set mined by GAIA as the given pattern set, and perform D&D to see whether it can improve the performance of classification. We compare D&D and GAIA on both chemical datasets and protein datasets.

Fig. 14 shows the classification accuracy for D&D and GAIA on balanced chemical datasets. Our method obtains higher accuracy on total 11 chemical datasets, which validates the effectiveness of our D&D criteria on even small and incomplete pattern set. We notice that the improvement is not as significant as on the frequent pattern set compared to Logratio. The underlying reasons are: the patterns mined by GAIA are not complete, and some important patterns miss, which limit the classification accuracy of D&D.

Fig. 15 shows the performance for D&D and GAIA on the unbalanced chemical datasets. D&D improves the accuracy on 10 out of 11 datasets. Fig. 16 shows the running time by D&D and GAIA on 11 unbalanced chemical datasets. The time of D&D includes GAIA's mining time, D&D's feature selection time and the classification time. The result shows that we can improve the classification accuracy with a small overhead on feature selection by D&D.

Fig. 17 shows the classification accuracy for D&D and GAIA on the balanced protein datasets. Our method gets better accuracy on 15 out of 16 datasets. The improvement is more significant compared to that on chemical datasets.

Fig. 18 shows the performance for D&D and GAIA on the unbalanced protein datasets. D&D wins on 14 out of 16 datasets. Fig. 19 shows the running time for D&D and GAIA on the unbalanced protein datasets. Again, the time of D&D includes GAIA's mining time, D&D's feature selection time and the classification time. D&D only adds a small overhead on GAIA for feature selection, but improves the classification accuracy substantially.

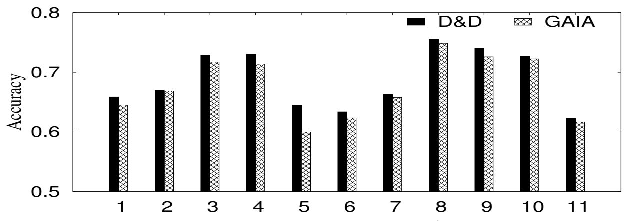


Figure 14: Accuracy Comparison for D&D and GAIA (Balanced Chemical Datasets)

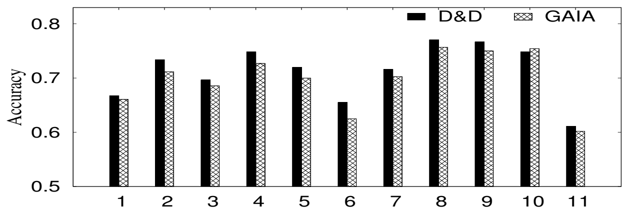


Figure 15: Accuracy Comparison for D&D and GAIA (Unbalanced Chemical Datasets)

8. RELATED WORK

In the literature, there have been a number of studies on discovering or selecting different kinds of patterns to build a graph classification model. One traditional way is to mine frequent patterns first, then select interesting features, and finally build a classification model [4] [5] [1]. Recently people have proposed mining algorithms that directly mine the most discriminative features efficiently [20, 17, 18, 16, 11, 12]. Yan et al. [20] proposed an efficient algorithm, called LEAP, for mining the most discriminative subgraph. The algorithm focuses on searching dissimilar subgraphs using structural leap search and achieves significant speedup over the widely used branch-and-bound approach. [17] proposed an iterative subgraph mining method based on partial least squares regression (PLS), combined with a weighted pattern mining algorithm. [18] proposed a feature selection approach on frequent subgraphs, called CORK, that optimizes a submodular quality criterion and integrates the criterion into gSpan for subgraph mining. Ranu and Singh [16] proposed GraphSig, a scalable method to mine significant (measured by p-value) subgraphs based on a feature vector representation of graphs. The first step in the mining process is to convert each graph into a set of feature vectors where each vector represents a region within the graph. Prior probabilities of features are computed empirically to evaluate statistical significance of patterns in the feature space. Then frequent subgraph mining techniques can be used to mine significant patterns in a scalable manner. [11] designs a new classification method based on pattern co-occurrence to derive graph classification rules. [12] proposes a subgraph mining method, GAIA, which employs a novel subgraph encoding approach to support an arbitrary subgraph pattern exploration order and explores the subgraph pattern space using evolutionary computation. In this manner, GAIA is able to find discriminative subgraph patterns much faster than other algorithms. Most of the above methods do not rigorously measure the feature overlap. Different from the above studies, [24] proposes to use graph metrics, such as non-tree density, degree distribution, diameter, global clustering coefficient, etc., as features for graph classification.

Graph pattern mining has been widely studied and can be catego-

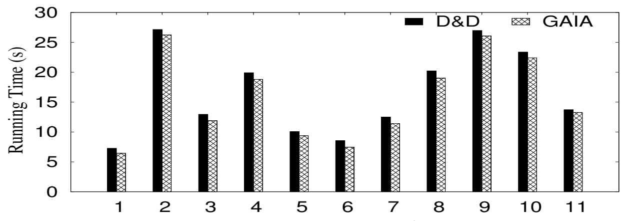


Figure 16: Runtime Comparison for D&D and GAIA (Unbalanced Chemical Datasets)

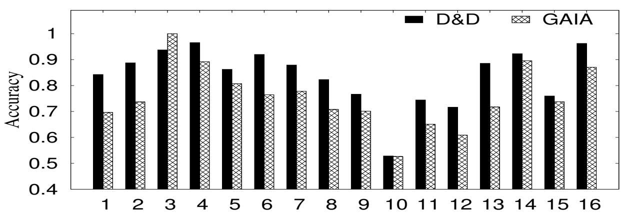


Figure 17: Accuracy Comparison for D&D and GAIA (Balanced Protein Datasets)

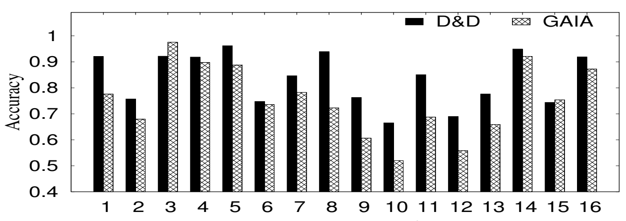


Figure 18: Accuracy Comparison for D&D and GAIA (Unbalanced Protein Datasets)

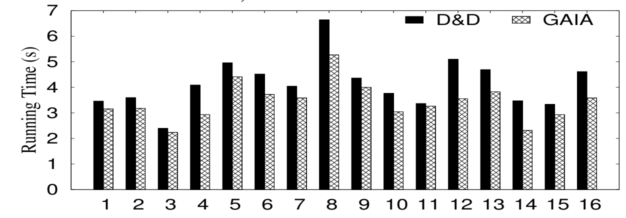


Figure 19: Runtime Comparison for D&D and GAIA (Unbalanced Protein Datasets)

rized in three types: exact graph mining such as gSpan [21], FFSM [9], Gaston [14] and FSG [13] ; approximate graph mining such as SUMMARIZE-MINE [3], TFP [7], Monkey [23] and sampling method [8]. Other methods include maximal pattern mining [10] and closed pattern mining [22].

Feature selection is a critical step for building accurate classification model [19, 6, 15, 4]. In conventional classification tasks on high-dimensonal data, feature selection is a well studied problem in the literature, which aims to select highly relevant and less redundant features, such as CMIM [6], mRmR [15], and MMRFS [4]. However, such feature selection methods have limit power when they are directly applied to graph database. This is because these methods treat each feature as an independent unit and evaluate their relationship only by their distributions over binary vectors. While in graph database, features have complicated relationships in terms of the topological structures and the locations. Moreover, a pattern has an exponential number of subgraphs, and the patterns are not uniformly distributed. This leads to a huge redundancy and complex overlapping of patterns.

9. CONCLUSION

In this paper we study graph classification by designing a diversified discriminative feature selection approach. Our proposed score can reduce the overlap between selected features by considering the embedding overlaps in the graphs. We analyze the properties of our score measure and show that the selected features make the positive and negative graphs more separable. Experimental results show that we can achieve a higher classification accuracy compared with the state-of-the-art graph classification methods with our proposed diversified discriminative feature selection method.

Acknowledgments The work was supported by grants of the Research Grants Council of the Hong Kong SAR, China No. 419109, 411310, 411211 and 418512.

10. REFERENCES

[1] D. Bandyopadhyay, J. Huan, J. Liu, J. Prins, J. Snoeyink, W. Wang, and A. Tropsha. Structure-based function inference using protein family-specific fingerprints. *Protein Science*, 15(6):1537–1543, 2006.

[2] 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.

[3] C. Chen, C. X. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han. Mining graph patterns efficiently via randomized summaries. *PVLDB*, 2(1):742–753, 2009.

[4] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725, 2007.

[5] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *TKDE*, 17(8):1036–1050, 2005.

[6] F. Fleuret. Fast binary feature selection with conditional mutual information. *JMLR*, 5:1531–1555, 2004.

[7] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, pages 211–218, 2002.

[8] M. A. Hasan and M. J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009.

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

[10] J. Huan, W. Wang, J. Prins, and J. Yang. Spin: mining maximal frequent subgraphs from graph databases. In *KDD*, pages 581–586, 2004.

[11] N. Jin, C. Young, and W. Wang. Graph classification based on pattern co-occurrence. In *CIKM*, pages 573–582, 2009.

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

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

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

[15] H. Peng, F. Long, and C. H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *TPAMI*, 27(8):1226–1238, 2005.

[16] S. Ranu and A. K. Singh. Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *ICDE*, pages 844–855, 2009.

[17] H. Saigo, N. Krämer, and K. Tsuda. Partial least squares regression for graph mining. In *KDD*, pages 578–586, 2008.

[18] 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*, 2009.

[19] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *NIPS*, pages 668–674, 2001.

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

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

[22] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *KDD*, pages 286–295, 2003.

[23] S. Zhang, J. Yang, and V. Cheedella. Monkey: Approximate graph mining based on spanning trees. In *ICDE*, pages 1247–1249, 2007.

[24] L. Zhu, W. K. Ng, and S. Han. Classifying graphs using theoretical metrics: A study of feasibility. In *DASFAA*, 2011.