Structure Feature Selection For Graph Classification

Hongliang Fei, Jun Huan
Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66047-7621, USA
{hfei, jhuan}@ittc.ku.edu

ABSTRACT

With the development of highly efficient graph data collection technology in many application fields, classification of graph data emerges as an important topic in the data mining and machine learning community. Towards building highly accurate classification models for graph data, here we present an efficient graph feature selection method. In our method, we use frequent subgraphs as features for graph classification. Different from existing methods, we consider the spatial distribution of the subgraph features in the graph data and select those ones that have consistent spatial location.

We have applied our feature selection methods to several cheminformatics benchmarks. Our method demonstrates a significant improvement of prediction as compared to the state-of-the-art methods.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Data Mining, Classification, Feature Selection

1. INTRODUCTION

Knowledge discovery in complex data is an essential problem in data mining. Graph, a discrete structure, has been widely applied in diverse areas such as Cheminformatics, Bioinformatics, social network analysis, and database management as a tool to model a wide range of data. A major difficulty in mining graph data lies in the high complexity, which is from graph's complex structure and the intrinsic high dimensionality of graphs. The objective of this paper is to derive an automated way to construct a low-dimensional vector representation for graphs through developing a highly effective feature selection methods. Finding a proper vector representation of graphs may lead to more accurate models, reduced computational time, and better explanations of the real relationship between graphs and graph labels in classification, and hence worth a careful investigation.

Current solutions for feature selection problems can be roughly divided into two categories: feature extraction and feature selection [1]. Principle Component Analysis (PCA) projects data to an eigenvector space to reduce the dimensionality and hence to obtain a small number of features [13, 19]. Similar methods for feature extraction include Linear Discriminative Analysis LDA [29], Local Linear Embedding LLE [21] and Isomap [25]. Using Kernel PCA, investigators have designed algorithms to embed a graph to a vector space and achieved good empirical results in classification [22, 26].

Traditional feature filtering methods select individual features whose distribution correlates the distribution of the class labels. Such methods include term frequency thresholding, mutual information, information gain, χ^2, and Pearson Correlation as studied in [28]. In contrast to filtering method, which do not consider the dependency between features and may select redundant features, wrapper methods search through the feature subset space and identify highly informative features by using a classifier to score the subsets of features [17, 18].

Adapting existing feature extraction and feature selection methods to graph data is non trivial. First, graph are semi-structured data. There is no obvious choices of features in graphs to start feature selection methods. Second, graph kernel functions map graphs to a Hilbert space implicitly and thus avoid the problem of direct feature extraction. Though theoretic appealing, limited progresses have been made in reality in applying graph kernel functions to extract useful features in graphs. This is due to several reasons: (i) design a graph kernel function is not easy, (ii) the connection of kernel space and the original graph space is not clear and (iii) it is hard to explain the physical meaning of the identified features using kernel PCA techniques.

For special cases, investigators have identified that adapting existing feature selection methods to graph data is trivial. For example, frequent subgraph mining is a technique that extracts high frequent subgraphs in a group of graphs. Once such graphs are created, we may treat each subgraph as a feature and transform a graph to a vector, indexed by the identified subgraphs, in the related feature space. Any existing feature selection methods for Euclidian space are

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

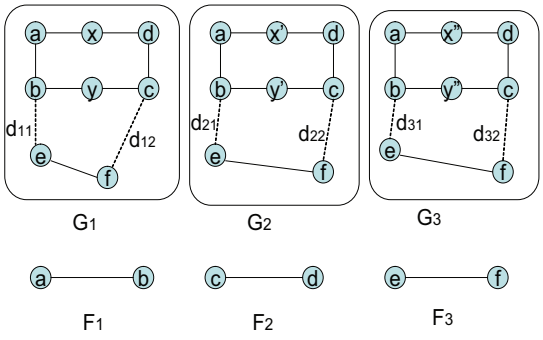

Figure 1: Spatial distribution of three frequent subgraphs in a graph database

applicable to select highly informative features in the subgraph feature space.

We argue that neither a simple postprocessing nor a direct adaption of the kernel PCA method identifies the intrinsic structure features of graph data completely. Our intuition is that the spatial distribution of subgraph features carries important information regarding the importance of the feature in a classification task. We illustrate the point with the following example:

In Figure 1, we show a graph database with three graphs G_1, G_2, G_3 and three subgraph features F_1, F_2, F_3. Subgraph F_1 and F_2 occur in every graph with a consistent relative spatial distribution. F_3 also occurs in every graph, but In contrast to F_1 and F_2, has quite different spatial distribution as compared to F_1 and F_2. In regular feature selection, F_1, F_2, and F_3 occur in the same set of graphs and hence may be perceived to have the same classification power. Clearly this is not the case in this example. Based on intuition, we have designed an integrated approach of two existing approaches: graph kernels and subgraph mining by designing a feature selection method working on graph spectrum kernels to gain deeper understanding of graph data. Our comprehensive experimental study of the designed algorithms using real-world data sets demonstrated the power of the novel feature selection method.

1.1 Problem Statement

Given a set of graphs $\mathcal{G}$, each graph in $\mathcal{G}$ has an associated labels c, and a set of subgraphs $\mathcal{F}$ extracted from $\mathcal{G}$, the **graph feature selection problem** is to select a subset of features $\mathcal{F}_s \subset \mathcal{F}$, to obtain better classification accuracy than using $\mathcal{F}$ only for the graph data set.

1.2 Related Work

Extracting features, in the form of subgraphs, from graph data has been well studied in graph database mining methods. The goals of such methods extract highly informative subgraphs from a set of graphs. Typically some filtering criteria are applied, among those the most widely used is the frequency of a subgraph. For example, Huan *et al.* develop a depth-first search algorithm: Fast Frequent Subgraph Mining (FFSM) [11]. This algorithm identified all connected subgraphs that occurs in a large fraction of graphs in a graph data set. Additional filtering criteria are also developed, such as coherence [12], closeness [27], maximal size [12], density [9] among many others. Majority of the frequent subgraph feature extraction methods are unsupervised, meaning that there is no class labels information available (or such information are deliberately ignored) with a few exceptions. For example, an odd ratio is used to select subgraphs that is highly informative to build classifier in [10].

Many existing feature selection methods for vector space are supervised, determining the relevance of a feature through computing the correlation of feature value distribution and class label distribution. Traditional feature filtering methods select features independent of any classifier. In contrast to filtering method, which do not consider the dependency between features and may select redundant features, wrapper methods search through the subset space and identify highly informative features by using a classifier to evaluate the classification power of subsets of features and identify optimal subsets [17].

Kernel methods are now widely used in supervised learning and feature selection as well. For example, in the method of Support Vector Machine Recursive Feature Elimination (SVM_RFE) [7], SVM_RFE selects features via a greedy backward feature elimination. SVM_RFE first builds a linear classifier, then uses the weight vector of the hyperplane constructed by the training samples to rank features. During each iteration, lower ranked features are removed and new hyperplane is constructed and so on so forth. The limitation of SVM_RFE is that it works only with linear kernel.

Spectral feature selection [30], as a filtering method, explored an uniformed frame for feature selection in both unsupervised and supervised learning. It first constructed an object graph, where each node is corresponding to an object of training data; then ranked features using graph spectral decomposition and selected a subset of features based on their rank. Since spectral feature selection is a filtering methods, the feature dependency information is ignored. Cao *et al.* recently developed a method for feature selection in the kernel space rather than the original feature space based on Maximum Margin concept. Without tracing back into original feature space, they could select features in Kernel space.

Maximum Margin Feature selection (MMRFS) [3] is a wrapper method. In this method MMRFS uses information gain to weigh the correlation between each feature and class labels, then selectes features with less redundancy and covering new training samples.

Though feature selection have been developed for a long time, none of the existing method considers the special characteristics of graph data and hence may not provide the optimal results for graph feature selection. The objective of this paper is to develop a highly effective graph feature selection methods.

2. BACKGROUND

Here we introduce basic notations for graph, frequent subgraph mining and graph kernel functions.

2.1 Graph Theory

A *labeled graph* G is described by a finite set of nodes V and a finite set of edges $E \subset V \times V$. In most applications, a graph is labeled, where labels are drawn from a label set λ. A labeling function $\lambda : V \cup E \rightarrow \Sigma$ assigns labels to nodes and edges. In *node-labeled graphs*, labels are assigned to nodes only and in *edge-labeled graphs*, labels are assigned to edges only. In *fully-labeled graphs*, labels are assigned to

nodes and edges. We may use a special symbol to represent missing labels. If we do that, node-labeled graphs, edge-labeled graphs, and graphs without labels are special cases of fully-labeled graphs. Without loss of generality, we handle fully-labeled graphs only in this paper. We do not assume any structure of label set Σ now; it may be a field, a vector space, or simply a set.

Following convention, we denote a graph as a quadruple $G = (V, E, \Sigma, \lambda)$ where V, E, Σ, λ are explained before. A graph $G = (V, E, \Sigma, \lambda)$ is a *subgraph* of another graph $G' = (V', E', \Sigma', \lambda')$, denoted by $G \subseteq G'$, if there exists a 1-1 mapping $f : V \rightarrow V'$ such that

- for all $v \in V, \lambda(v) = \lambda'(f(v))$
- for all $(u, v) \in E, (f(u), f(v)) \in E'$
- for all $(u, v) \in E, \lambda(u, v) = \lambda'(f(u), f(v))$

In other words, a graph is a subgraph of another graph if there exits a 1-1 node mapping such that preserve the node labels, edge relations, and edge labels.

The 1-1 mapping f is a *subgraph isomorphism* from G to G' and the range of the mapping f, $f(V)$, is an *embedding* of G in G'.

2.2 Frequent Subgraph Mining

Given a graph database $\mathcal{G}$, the support of a subgraph G, denoted by sup_G, is the fraction of the graphs in $\mathcal{G}$ of which G is a subgraph, or:

$$sup_G = \frac{|G^{'} \in GD | G \subseteq G^{'}|}{|GD|}$$

Given a user specified minimum support threshold *min_sup* and graph database $\mathcal{G}$, a *frequent subgraph* is a subgraph whose support is at least *min_sup* (i.e. $sup_G \geq min\_sup$) and the *frequent subgraph mining problem* is to find all frequent subgraphs in $\mathcal{G}$.

In this paper, we use frequent subgraph mining to extract features in a set of graphs. Each mined subgraph is a feature. Each graph is transformed to a feature vector indexed by the extracted features with values indicate the presence or absence of the feature as did in [11]. We use binary feature vector as contrast to occurrence feature vector (where the value of a feature indicates the number of occurrences of the feature in an object) due to its simplicity. Empirical study shows that there is negligible difference between the two representations in graph classification.

2.3 Graph Kernel Function

Kernel functions are powerful computational tools to analyze large volumes of graph data [8]. The advantage of kernel functions is due to their capability to map a set of data to a high dimensional Hilbert space without explicitly compute the coordinates of the structure. This is done through a special function K. Specifically a binary function $K : X \times X \rightarrow \mathbb{R}$ is a *positive semi-definite* function if

$$\sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) \geq 0 \quad (1)$$

for any $m \in \mathbb{N}$, any selection of samples $x_i \in X$ ($i = [1, n]$), and any set of coefficients $c_i \in \mathbb{R}$ ($i = [1, n]$). In addition, a binary function is symmetric if $K(x, y) = K(y, x)$ for all $x, y \in X$. A symmetric, positive semi-definite function ensures the existence of a Hilbert space $\mathcal{H}$ and a map $\Phi : X \rightarrow \mathcal{H}$ such that

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (2)$$

for all $x, x' \in X$. $\langle x, y \rangle$ denotes an inner product between two objects x and y. The result is known as the Mercer's theorem and a symmetric, positive semi-definite function is also known as a Mercer kernel function [22], or *kernel* function for simplicity.

Several graph kernel functions have been studied. Recent progresses of graph kernel functions could be roughly divided into two categories. The first group of kernel functions consider the full adjacency matrix of graphs and hence measure the global similarity of two graphs. These include product graph kernels [6], random walk based kernels [15], and kernels based on shortest paths between pair of nodes [16]. The second group of kernel functions try to capture the local similarity of two graphs by counting the shared subcomponents of graphs. These include the subtree kernels [20], cyclic kernels [24], spectrum kernel [5], and recently frequent subgraph kernels [23]. In this paper, we focus on graph spectrum kernels where we use subgraph as features. The feature vector of a graph is the list of subgraph features that occurs in the graph and the kernel function of two graphs is the inner product of their feature vectors [5].

3. METHODOLOGY

Our structure based feature selection method has two steps: (1) feature extraction and (2) feature selection. In the feature extraction step, we mine frequent subgraphs in the training samples as features. We then apply a feature selection method, as discussed below, to select a small subset of features to build graph classification model.

Our hypothesis in feature selection is that the spatial distribution of subgraph features carries important information regarding the importance of the feature. To test our hypothesis, we have designed a feature selection method and have performed comprehensive experimental study using real-world data sets. Below we first present a feature selection framework. We then introduce the feature consistency map, which when combined with our feature selection framework, leads to a practical algorithm to select highly informative features.

3.1 Notation

In this paper, we use capital letters, such as G, for a single graph and upper case calligraphic letters, such as $\mathcal{G} = G_1, G_2, \ldots, G_n$, for a set of n graphs. We assume each graph $G_i \in \mathcal{G}$ has an associated class label c_i from a label set C. We use $\mathcal{F} = F_1, F_2, \ldots, F_n$ for a set of n features. Given a set of n features $\mathcal{F}$ and a graph G, we create a feature vector for G, denoted by $G^{\mathcal{F}}$, indexed by features in $\mathcal{F}$ and with values indicate whether the related feature is present (1) or absent (0) in the graph G. In other words, $G^{\mathcal{F}} = (G^{F_i})_{i=1}^{n}$ and

$$G^F = \begin{cases} 1 & \text{if } F \subseteq G \\ 0 & \text{otherwise} \end{cases}$$

3.2 Kernel-Target Alignment Framework

In this work we consider selecting features whose distribution is consistent with the distribution of the class labels. Towards that end, we compute the *object kernel matrix* ξ as defined below.

$$\xi = ((c_i == c_j))_{i,j=1}^{n} \tag{3}$$

where $(X == Y) = 1$ if $X = Y$ and otherwise 0. c_i is the class label of the ith object.

Desired features are those that are "consistent" to the object kernel matrix. The technical challenge here is how to measure the consistency. Recently Zhao & Liu proposed a method called spectral feature selection [30]. In the spectral feature selection method, the normalized Laplacian matrix $\mathcal{L}$ is computed as:

$$L = D - \xi; \mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{4}$$

Where D is the density matrix of ξ and is defined as $D = (d_{i,j})_{i,j=1}^{n}$ where

$$d_{i,j} = \begin{cases} \sum_{k=1}^{n} \xi_{i,k} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Given a feature $F = (f_i)_{i=1}^{n}$ where f_i is the value of the feature in the ith object, the value $< F^T, LF^T >$ measures the consistency of the feature F to the object kernel matrix. The smaller the value is, the most consistent the feature vector F is. Based on this measurement, Zhao & Liu proposed a filtering method by selecting the top k features that best consistent with the object kernel matrix ξ.

Different from their method, we adopt the Kernel-Target Alignment framework, originally proposed for automated kernel selection [4]. The advantage of the new framework is that (i) the new framework handles the dependency of features and hence may select better feature subset and (ii) the new framework allows us to use feature kernel and hence may handel non-linear correlation between a feature and class labels. The kernel-target Alignment method is discussed below.

Given a feature F, we define a feature kernel matrix S_F as

$$S_F = (K(G_i^F, G_j^F))_{i,j=1}^{n} \tag{5}$$

In the formula, K can be any kernel function. For simplicity in our experimental study we use linear kernel $K(X, Y) = X \times Y$.

With the feature kernel matrix and object kernel matrix, we use the following formula to measure whether the feature kernel is "consistent" with the object kernel. Toward that end, we introduce a binary function $\cdot : M \times M \rightarrow \mathbb{R}$ to compute the inner product of two matrices as

$$X \cdot Y = trace(X^T \times Y) \tag{6}$$

where M is the set of all n by m matrices.

Clearly the $\cdot$ function is a bilinear form, which means that it has the following properties:

$$(X_1 + X_2) \cdot Y = X_1 \cdot Y + X_2 \cdot Y$$
$$X \cdot (Y_1 + Y_2) = X \cdot Y_1 + X \cdot Y_2$$
$$(\lambda X) \cdot Y = \lambda (X \cdot Y) = X \cdot (\lambda Y)$$

In addition, $\cdot$ is positive ($X \cdot X = 0 \Rightarrow X = 0$) and symmetric ($X \cdot Y = Y \cdot X$) and hence a inner product function.

Based on the function $\cdot$, we define the similarity between two matrices is the inner product of the two matrices X and Y, normalized by the norm of the X and Y, or

$$\mathcal{S}(X, Y) = X \cdot Y / (||X|| \times ||Y||). \tag{7}$$

where $||X|| = \sqrt[2]{X \cdot X}$.

Geometrically the similarity function $\mathcal{S}$ measures the cosine value of the angle between two kernel matrices. This measurement is first used in [4] to automatically select kernel functions. We adapt it here for the purpose of feature selection. Before we proceed to our feature selection method, we present an important data structure, which we call feature consistency map.

3.3 Feature Consistency Map

Different from the kernel target alignment framework, whose goal is to measure the consistency between a feature and a class label. Here we develop a way to measure the consistency in a set of features without the class label information. Such measurement is clearly unsupervised, meaning that we do not take class label distribution into consideration. Our main source of information is the spatial distribution of subgraph features. To quantify the information, we propose a data structure called feature consistency map, which is built upon the following three concepts: embedding distance, embedding profiles, feature consistency, as detailed below.

DEFINITION 3.1. *(**Embedding Distance**) Given two frequent subgraph features F_i and F_j and their embeddings of e_i e_j in a graph G, the embedding distance, denoted by $d^G(e_i, e_j)$ is defined as:*

$$d^G(F_i, F_j) = \frac{\sum_{u \in e_i} \sum_{v \in e_j} d(u, v)}{|e_i| \times |e_j|} \tag{8}$$

Where $d(u, v)$ is the shortest distance between the node u and the node v.

EXAMPLE 3.1. *In Figure 2, subgraph feature F_1 has an embedding $\{a, b\}$ in G_1. F_2 has an embedding $\{c, d\}$ in G_1. Hence, the embedding distance between pattern F_1 and F_2, denoted by $d^{G_1}(F_1, F_2)$ is $\frac{2+3+3+2}{2*2}$=2.5.*

It is possible that one frequent pattern has several embeddings in a graph. We illustrate the procedure we use to compute embedding distance with multiple embeddings in the appendix.

DEFINITION 3.2. *(**Embedding Profile**) Given a subgraph feature G, the **embedding profile** of the subgraph G in a set of graphs $\mathcal{G}$, denoted by $\mathcal{P}_{\mathcal{G}}G$, is the set of embedding of G in $\mathcal{G}$, or:*
$\mathcal{P}_T G = \{e \mid e \text{ is an embedding of } G \text{ in } G' \in \mathcal{G}\}$

DEFINITION 3.3. *(**Feature Consistency Relationship**) Given two frequent patterns f_i and f_j in a set of graph $\mathcal{G}$, let $d^{\mathcal{G}} = (d^G(f_i, f_j))_{G \in \mathcal{G}}$ denote the list of embedding distances computed between f_i and f_j, f_i and f_j are **consistent** if the variance of embedding distance vector $d^{\mathcal{G}}$ is less than some threshold max_var.*

EXAMPLE 3.2. *In Figure 1, $d^{G_1}(F_1, F_2) = d^{G_2}(F_1, F_2) = d^{G_3}(F_1, F_2) = 2.5$. The variance of the distance vector $d^{\mathcal{G}}$ is 0 and hence F_1 and F_2 is consistent. Assuming the shortest path connecting F_1 and F_3 in different graphs have dramatically different lengths, we conclude that F_1 and F_3 are not consistent.*

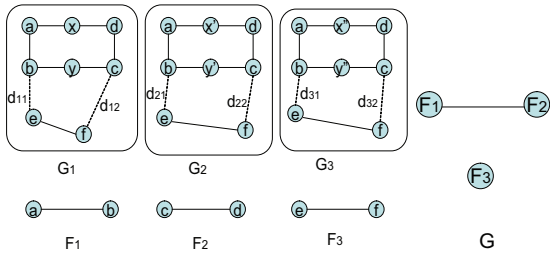

Figure 2: Left: Three subgraphs and three embeddings of the subgraphs. This figure is duplicated from Figure 1 for clarity. Right: The feature consistency map for the three subgraphs shown in Figure 1

Consistency measures the stableness of the spatial distribution of two frequent subgraphs. We use this criteria to select features with a stable spatial distribution.

DEFINITION 3.4. *(**Feature Consistency Map**) A **feature consistency map** is a graph $G = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ and arc set $\mathcal{E}$ are specified below:*

- *Each vertex in $\mathcal{V}$ represents a feature*
- *Two vertices are connected by an arc in $\mathcal{E}$ if the two vertices are consistent (as defined in the feature consistency relationship)*

To avoid confusion of the feature consistency map and a regular graph that we specified before, we use vertex and arc to denote commonly used term "node" and "edge".

The computation of the feature consistency map for a feature set is straightforward: for each pair of features, we collect their embedding profiles and then decided whether the two features are consistent. Based on the consistency relationship, we could easily construct the feature consistency map.

We summarize what we have discussed in the following pseudo code of the function FCM for constructing the feature consistency map. In the pseudo code, $\mathcal{F}$ is a set of subgraph features and $\mathcal{G}$ is the set of training graph samples where the subgraph features are extracted.

Algorithm 1 FCM($\mathcal{F}, \mathcal{G}$)

```
1: V = F, E = ∅
2: for G, G' ∈ F do
3:    L ← P_G G
4:    L' ← P_G G'
5:    if L and L' is consistent then
6:       E ← E ∪ (G, G')
7:    end if
8: end for
9: return G = (V, E)
```

In what follows, we introduce two feature selection methods. The first one is a feature filtering method; it selects features individually. The second one is a wrapper methods; it selects features in the context of selected features.

3.4 Feature Ranking and Forward Structure Based Feature Selection

Once we compute the feature consistency map G, we use a simple way to rank the features by sorting the features according to their degree (number of edges incident on the feature) in G in descending order. After sorting features according to their node degree, we select top k features. We call this method SFS_Filtering.

In forward structure based feature selection, we sort the features using the same procedure as we do in the feature filtering method. Different from the feature filtering method, we use the Equation 7 and select features in the context of selected ones. Specifically, we evaluate the similarity between the resulting kernel matrix (may contain several features) and the object kernel matrix and make sure when we select a feature, the similarity value monotonically increases.

The following is the algorithm for forward selection, where S_F denotes the feature kernel matrix, ξ is the object kernel matrix, and $\mathcal{F}$ is a set of subgraph features as we discussed before.

Algorithm 2 SFS_FS($\mathcal{F}, \xi$)

```
1: F_s = ∅, T_0 = 0
2: sort F according to the node degree in the related feature
   consistency map
3: n ← |F|
4: for i = 1, ..., n do
5:    F' ← F_s ∪ F_i
6:    T ← S(S_F', ξ)
7:    if T > T_0' then
8:       F_s ← F'
9:       T_0 ← T
10:   end if
11: end for
12: return F_s
```

The adaptation of the previously mentioned algorithm to a backward selection algorithm is straightforward. We start with the full set of feature and eliminate one by one. In the elimination process, we make sure the similarity of the feature kernel matrix and the object kernel matrix is increasing. Otherwise, we keep the feature. Though conceptually similar, our empirical evaluation shows that backward feature selection is not as good as forward feature selection.

Finally, we notice that we may augment a weight to each node in the feature consistency map. A straightforward way to assign a weight to a node is to compute the Pearson's Correlation Coefficient between the feature and the class labels in the training data set. Many other choices are available, such as mutual information [28] and odd ratio [12]. It is difficult to enumerate all possible choices and we use Pearson Correlation because empirically it performs well. Unless stated otherwise, the feature consistency map that we use in this paper is always node-weighted where the weight of the node is computed using Pearson's Correlation Coefficients.

4. EXPERIMENTAL STUDY

We have performed a comprehensive study of the performance of our feature selection method using 5 chemical structure graph datasets. We have compared our method with 4 state-of-the-art feature selection methods: SVM Recursive Feature Elimination (SVM_RFE) [7], Spectral Fea-

ture Selection [30], Log Odd Ratio Feature selection (LORFS) [10] and Maximum Margin Feature selection (MMRFS) [3].

For each data set, we used the FFSM algorithm [11] to extract frequent subgraph features from the data sets and used the LibSVM package [2] to train a Support Vector Machine (SVM) for classification. We measured the classification performance of our feature selection method and compared ours with those from state-of-the-art methods using cross validation.

4.1 Data Sets

We use five data sets from drug virtual screening experiments [14]. In each data set, the target values are drugs' binding affinity to a particular protein. For each protein, the data provider selected 50 chemical structures that clearly bind to the protein ("active" ones). The data provider also listed chemical structures that are very similar to the active ones (judged with domain knowledge) but clearly do not bind to the target protein. This list is known as the "decoy" list. We randomly sampled 50 chemical structures from the decoy list. See [14] for further details regarding the nature of the data set. To reiterate, each of these 5 data sets contains 100 compounds with 50 positives and 50 negatives.

We followed the same procedure [11] to use a graph to model a chemical structure: a node represents an atom and an edge represents a chemical bond. We removed Hydrogen atoms in our graph representation of chemicals, as commonly done in the cheminformatics field. The characteristics of the data set is shown in Table 2.

Table 2: Data set: the symbol of the data set. S: total number of samples in the data set. P: total number of positive samples, N: total number of negative samples, $\overline{V}$: average number of nodes in the data set, $\overline{E}$: average number of edges in the data set

Data set	S	P	N	$\overline{V}$	$\overline{E}$
A1A	100	50	50	26	28
CDK2	100	50	50	22	25
COX2	100	50	50	22	24
FXa	100	50	50	27	29
PDE5	100	50	50	26	28

4.2 Experimental Protocol

For each data set, we mined frequent subgraphs using the FFSM algorithm [11] with $min\_support = 50\%$ and with at least 5 nodes and no more than 10 nodes. We then treated each subgraph as a feature. We create a binary feature vector for each graph in the data set, indexed by the mined subgraphs, with values indicate the existence (1) or absence (0) of the related features. All feature selection methods that we compared with are based on the same feature sets.

As indicated before, we compared 4 feature selection methods. To have a fair comparison, we select a fixed number of k features using each method and compare the classification accuracy of the selected features. In our experiments, we set $k = 25$. Empirical study shows that there is no significant change if we replace the fixed value 25 with a wide range of values.

To compute the feature consistency map that is used in the Pattern_SFS method, we set max_var to be 0.5. For the MMRFS method, we used the default parameter (coverage threshold δ=1 and $min\_sup = 0.5$). We implement our own version of the spectral feature selection method. This is a filtering method with no additional parameters. We use the SVM_RFE executable as included in the spider machine learning toolbox downloaded from `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`.

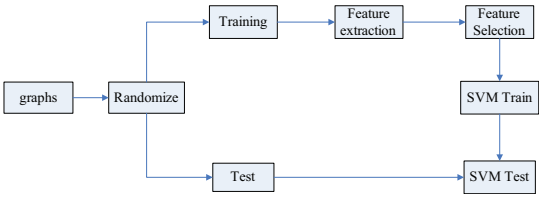

Figure 3: Experimental workflow for a single cross-validation trial.

We use LIBSVM perform training and testing. To have a fair comparison we use RBF kernel with default parameters (C=1, γ=0.5) in all the experiments we run. We use standard 5-fold cross validation to derive training and testing samples. For a cross validation, we compute precision as (TP/(TP+FP)), recall as (TP/(TP+FN)), and accuracy as (TP+TN/S) where TP stands for true positive, TN stands for true negative and S stands for the total number of samples. For each data set, we repeat the 5-fold cross validation 10 times and report the average precision, recall, and accuracy. Standard deviation of classification results is not small, in a range of 5% to 10%. We performe all of our experiments on a desktop computer with a 3Ghz Pertium 4 processor and 1 GB of RAM.

Figure 3 gives an overview of our experimental set up.

4.3 Experimental Results

4.3.1 Performance Comparison

In this section, we present the performance of our method compared with four additional methods: Spectral feature selection, SVM_RFE, LORFS and MMRFS. The accuracy is shown in Figure 5 and the precision, recall are as shown in Table 1.

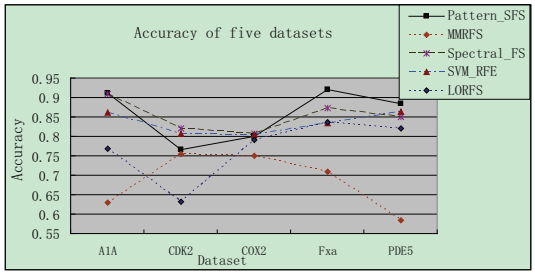

Figure 5: Comparing the classification accuracy of 5 feature selection methods on 5 data sets.

Pattern_SFS outperforms MMRFS, LORFS in all the 5 datasets, outperforms the SVM_RFE method and the spectral feature selection method 3 out of the 5 data sets. Overall, Pattern_SFS is the best methods in the group in 3 out

Table 1: Average Precision and Recall of four methods on 5 data sets. Stars (*) denote the highest precision or call among all competing methods for a dataset.

	Pattern_SFS		MMRFS		Spectral_FS		SVM_RFE		LORFS	
Data set	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
A1A	0.890*	0.939	0.652	0.842	0.879	0.951*	0.857	0.882	0.811	0.706
CDK2	0.843	0.707	0.945*	0.613	0.875	0.724	0.819	0.809*	0.783	0.320
COX2	0.884	0.700	0.825	0.662	0.932	0.670	0.812	0.804*	0.973*	0.600
FXa	0.932	0.904	0.667	0.958*	0.899	0.845	0.838	0.823	0.962*	0.691
PDE5	0.901*	0.859*	0.702	0.675	0.881	0.823	0.867	0.847	0.832	0.800

Table 3: Classification accuracy of different implementations of the structure based feature selection methods.

Data set	Pattern_All	SFS_Filter	SFS_FS	SFS_BE	SFS_Filter_NPC	SFS_FS_NPC	PCCS
A1A	0.626	0.809	0.912*	0.488	0.812	0.821	0.898
CDK2	0.668	0.679	0.766	0.535	0.666	0.776	0.850*
COX2	0.756	0.817*	0.802	0.475	0.783	0.793	0.788
FXa	0.714	0.892	0.921*	0.576	0.891	0.918	0.865
PDE5	0.602	0.904*	0.885	0.574	0.902	0.888	0.849

of the 5 tested data sets. The results confirm our hypothesis that considering the spatial distribution of subgraph features leads to better selection of discriminative features.

4.3.2 Comparison of Variations of Structure Based Feature Selection

Here we compare seven variations of the basic Structure Based Feature Selection methods. These are:

- forward feature selection (SFS_FS),
- backward feature elimination (SFS_BE),
- filtering (SFS_Filter),
- forward feature selection with un-weighted feature consistency map (SFS_FS_NPC),
- filtering with un-weighted feature consistency map (SFS_Filter_NPC).

In addition, we show results without any feature selection (Pattern_all) and results with features selected by Pearson Correlation coefficient selection (PCCS). The results are shown in Table 3. From the table, we find that the performance of PC weighted forward selection is the best overall. In this paper, we use PC weighted forward selection to compare with other state-of-the art methods.

We observe that Pattern_All often achieves the worst result, which demonstrates that redundant features will result in over-fitting problem and diminish the classification accuracy. Although PCCS takes correlation between a single feature and label, it neglects dependence of features and hence usually do not select the optimal feature subsets.

4.3.3 Method Robustness With Parameters

Since we have several parameters in the feature extraction and feature selection process, we evaluate the robustness of our method by changing different parameter values. In the following study, we have singled out a single data set (the Fxa data set) and test the robustness of our method using this data set by varying three parameters: the *min_sup* in frequent subgraph feature extraction process, the total number of selected features (k), and the *max_var* parameter that is used to derive the feature consistency map.

On the left part of Figure 4, we show the result by changing *min_sup* from 0.25 to 0.50 on FXa dataset with $\#feature s = 25$ and *max_var*=0.5. It is obvious that our method remains stable with variant *min_sup*.

The middle part of Figure 4 indicates results on FXa data set by varying the number of selected features from 1 to 100. From the result, the classification accuracy remains stable within a wide range and the best accuracy can be obtained around 20 to 50.

We fix *min_sup*=30% and #features=25 to test whether the accuracy remains stable with different *max_var*. In right most of Figure 4, *max_var* is changed from 0.125 to 1.25, the accuracy is around 92.7% with standard deviation 0.6%.

Overall, our structure based feature selection method is effective and achieves good accuracy within a wide range minimum support, maximum variance and the number of selected features. Since feature selection can be viewed as a data preprocessing step, any other feature selection method can be combined with our framework and our method is applicable to any current start-of-art classifiers.

5. CONCLUSIONS

In this paper, we presented a novel feature selection method for graph classification. By ranking features based on their spatial distribution and their contributions to classification, we have designed a feature selection method (and several variations) called structure based feature selection method. Compared with current state-of-the-art methods as evaluated on 5 real world data sets, our method outperforms the 4 state-of-the-art methods on majority of the tested data sets. In the feature, we plan to extend structure feature selection to kernel space. Currently, our method can work in both unsupervised learning and supervised learning, hence another line of our future work is to do semi-supervised feature selection.

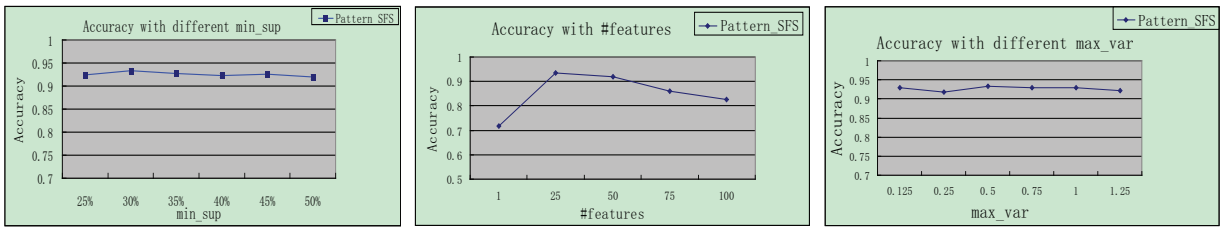

Figure 4: Left: Classification accuracy with varying minimal support. Middle: Accuracy with varying number of selected features. Right: Accuracy on varying *max_var* that is used to construct the feature consistency map.

Acknowledgments

This work has been supported by the Kansas IDeA Network for Biomedical Research Excellence (NIH/NCRR award #P20 RR016475) and a NIH grant #R01 GM868665.

6. REFERENCES

[1] B. Cao, D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Feature selection in a kernel space. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 121–128, New York, NY, USA, 2007. ACM.

[2] C. Chang and C. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*, 2007.

[4] N. Cristianini, J. Shawe-Taylor, and A. Elisseeff. On kernel-target alignment, 2001.

[5] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[6] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Sixteenth Annual Conference on Computational Learning Theory and Seventh Kernel Workshop*, 2003.

[7] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002 January.

[8] D. Haussler. Convolution kernels on discrete structures. *Technical Report UCSC-CRL099-10, Computer Science Department, UC Santa Cruz*, 1999.

[9] J. Huan, D. Bandyopadhyay, J. Snoeyink, J. Prins, A. Tropsha, and W. Wang. Distance-based identification of spatial motifs in proteins using constrained frequent subgraph mining. *in Proceedings of the IEEE Computational Systems Bioinformatics*, 2006.

[10] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. Mining protein family specific residue packing patterns >from protein structure graphs. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.

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

[12] J. Huan, W. Wang, A. Washington, J. Prins, R. Shah, and A. Tropsha. Accurate classification of protein structural families based on coherent subgraph analysis. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 411–422, 2004.

[13] I. Jolliffe. *Principal Component Analysis*. Springer; 2nd ed. edition, 1986.

[14] R. Jorissen and M. Gilson. Virtual screening of molecular databases using a support vector machine. *J. Chem. Inf. Model.*, 45(3):549–561, 2005.

[15] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. of the Twentieth Int. Conf. on Machine Learning (ICML)*, 2003.

[16] B. K.M. and K. H.-P. Shortest-path kernels on graphs. In *in Proc. of International Conference on Data Mining*, 2005.

[17] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[18] F. Li, Y. Yang, and E. P. Xing. From lasso regression to feature vector. In *Advances in Neural Information Processing Systems*, 2005.

[19] A. M. Martinez and A. C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[20] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *Technical Report, First International Workshop on Mining Graphs, Trees and Sequences*, 2003.

[21] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323 – 2326, 2000.

[22] B. Schölkopf and A. J. Smola. *Learning with Kernels*. the MIT Press, 2002.

[23] A. Smalter, J. Huan, and G. Lushington. Structure-based pattern mining for chemical compound classification. *Proceedings of the 6th Asia Pacific Bioinformatics Conference*, 2008.

[24] S. W. Tamas Horvath, Thomas Gartner. Cyclic pattern kernels for predictive graph mining. *SIGKDD*, 2004.

[25] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A

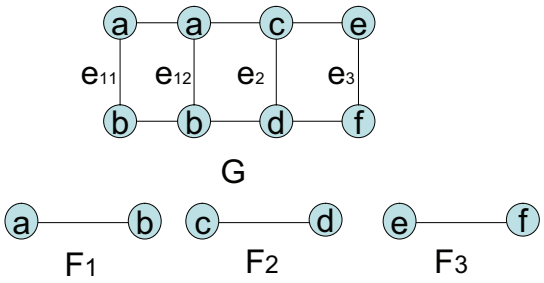

Figure 6: Frequent patterns with multiple embeddings in a graph

Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319 – 2323, 2000.

[26] S. Yan, D. Xu, B. Zhang, and H.-J. Zhang. Graph embedding: A general framework for dimensionality reduction. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 830–837, Washington, DC, USA, 2005. IEEE Computer Society.

[27] X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. *KDD'03*, 2003.

[28] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[29] H. Yu and J. Yang. A direct LDA algorithm for high-dimensional data Ů with application to face recognition. *Pattern Recognition*, 34(10):2067–2070, 2001.

[30] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

Appendix

In case when a frequent subgraph pattern has several embeddings in a graph, equation 8 is not applicable and we need to develop an approach to handle multiple embeddings. Toward that, we have developed a method to select the representative embedding when multiple embeddings in one graph happen, as detailed bellow.

DEFINITION 6.1. *(**Multiple Embedding Distance**) Given two frequent patterns F_i and F_j with multiple embeddings $e_{i1}, e_{i2}, ..., e_{im}$ of F_i and $e_{j1}, e_{j2}, ..., e_{jn}$ of F_j, we define the distance between an embedding e_{ik} and the pattern F_j in graph G as:*

$$dist^G(e_{ik}, F_j) = \frac{\sum_{l=1}^{n} d^G(e_{ik}, e_{jl})}{n}$$

where d^G is the embedding distance defined in Section 3.

EXAMPLE 6.1. *In Figure 6, subgraph feature F_1 has two embeddings in G denoted as e_{11} and e_{12}, F_2 has only one embedding e_2 and F_3 has e_3. The multiple embedding distance denoted by $dist^G(e_{11}, F_2)$ is $d^G(e_{11}, e_2) = \frac{10}{4}$. By the same token, we have $dist^G(e_{12}, F_2) = \frac{6}{4}$,$dist^G(e_{11}, F_3) = \frac{14}{4}$ and $dist^G(e_{12}, F_3) = \frac{10}{4}$.*

DEFINITION 6.2. *(**Multiple Embedding Distance Profile**) Given a set of features $\mathcal{F}$=$F_1, F_2, ..., F_n$ occurring in graph G and F_i with multiple m embeddings $e_{i1}, e_{i2}, ..., e_{im}$, the multiple embedding distance profile of e_{ik} $k = 1, 2, ...m$ is a vector denoted by $V^G(e_{ik})$ with each element representing the multiple embedding distance between e_{ik} and another feature:*

$$V^G(e_{ik}) = (dist^G(e_{ik}, F_j))_{F_j \in \mathcal{F} \& F_j \neq F_i}$$

EXAMPLE 6.2. *In Figure 6, we have $V^G(e_{11}) = (\frac{10}{4}, \frac{14}{4})$ and $V^G(e_{12}) = (\frac{6}{4}, \frac{10}{4})$.*

DEFINITION 6.3. *(**Representative Embedding**) Assume F_i has m embeddings in G, the representative embedding of F_i is the embedding that has the least distance summation to all the other patterns in this graph, formally denoted as $e_{i\hat{k}}$ where*

$$\hat{k} = \arg\min_k \sum V^G(e_{ik}) \ k = 1, 2, \ldots, m$$

EXAMPLE 6.3. *In Figure 6, since $V^G(e_{11}) = (\frac{10}{4}, \frac{14}{4})$, $V^G(e_{12}) = (\frac{6}{4}, \frac{10}{4})$ and , we select e_{12} as the representative embedding for F_1. Once e_{12} is selected, we do not consider e_{11} when constructing feature consistency map.*

The purpose of selecting representative embedding is to guarantee that one pattern is represented by one embedding in a graph so that the procedure of constructing feature consistency map becomes feasible. A representative embedding is like a centerpiece subgraph that avoids overlapping as much as possible.

There is another approach to select the representative embedding. For all the subgraph features that have multiple embeddings, we make an initial: randomly select an representative embedding for each feature with multiple embeddings. Then we take the average distance between each pattern and all the other patterns that occur in the graph, pick up the embedding whose embedding distance is the most close to the average value and repeat this procedure until converges.