Published in final edited form as:
IEEE/ACM Trans Comput Biol Bioinform. 2010 ; 7(2): 197–207. doi:10.1109/TCBB.2009.80.

GPD: A Graph Pattern Diffusion Kernel for Accurate Graph Classification with Applications in Cheminformatics

Aaron Smalter,
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS 66045

Jun (Luke) Huan,
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS 66045

Yi Jia, and
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS 66045

Gerald Lushington
Molecular Graphics and Modeling Laboratory, University of Kansas, Lawrence, KS 66045

Aaron Smalter: asmalter@ku.edu; Jun (Luke) Huan: jhuan@ku.edu; Yi Jia: jiayi@ku.edu; Gerald Lushington: glushington@ku.edu

Abstract

Graph data mining is an active research area. Graphs are general modeling tools to organize information from heterogeneous sources and have been applied in many scientific, engineering, and business fields. With the fast accumulation of graph data, building highly accurate predictive models for graph data emerges as a new challenge that has not been fully explored in the data mining community. In this paper, we demonstrate a novel technique called graph pattern diffusion (GPD) kernel. Our idea is to leverage existing frequent pattern discovery methods and to explore the application of kernel classifier (e.g., support vector machine) in building highly accurate graph classification. In our method, we first identify all frequent patterns from a graph database. We then map subgraphs to graphs in the graph database and use a process we call “pattern diffusion” to label nodes in the graphs. Finally, we designed a graph alignment algorithm to compute the inner product of two graphs. We have tested our algorithm using a number of chemical structure data. The experimental results demonstrate that our method is significantly better than competing methods such as those kernel functions based on paths, cycles, and subgraphs.

Index Terms

Graph classification; graph alignment; frequent subgraph mining

1 INTRODUCTION

Graphs are ubiquitous models that have been applied in many scientific, engineering, and business fields. For example, in finance data analysis, graphs are used to model dynamic

stock price changes [20]. To analyze biological data, graphs have been utilized in modeling chemical structures [33], protein sequences [41], protein structures [16], and gene regulation networks [17]. In web page classification, graphs are used to model the referencing relationship in HTML documents [47].

Due to the wide range of applications, development of computational and statistical frameworks for analyzing graph data has attracted significant research attention in the data mining community. In the past few years, various graph pattern mining algorithms have been designed [13], [14], [34], [36], [43], [46]. There are also many research efforts dedicated to efficiently searching graph databases [23], [32], [42], [44]. Much of the work on analyzing graph data previous to recent years involved unsupervised methods, where making predictions about graphs is usually not the goal. The research focus is well justified, since in order to make predictions of graph data we must have a large number of labeled training samples. Activities such as sample collection and sample labeling are time consuming and expensive.

With the rapid development of powerful and sophisticated data collection methods, there is a fast accumulation of labeled graph data. For example, many XML documents are modeled as trees or graphs and it is important to build classifiers for XML data [45]. As another example, natural language processing of sentences usually produces a tree (parsing tree) representation of a sentence. In many social science studies, building automated systems to classify sentences into several groups [25] is an important task.

What is especially interesting to us is the chemical classification problem in cheminformatics. Chemical structures have been studied using graph modeling for a long time [35]. With recently developed high throughput screening methods, the National Institute of Health has started an ambitious project called the Molecular Library Initiative aiming to determine and publicize the biological activity of at least a million chemical compounds each year in the next 5–10 years [2].

With the fast accumulation of graph data including class labels, *graph classification*, which we focus on in this paper, is an emergent research topic in the data mining community. Though classification has been studied for many years in data mining, graph classification is undeveloped and brings many new challenges. Below, we highlight a few of the new challenges.

In many existing classification algorithms [4], samples and their target values are organized into an object-feature matrix $X = (x_{i,j})$ where each row in the matrix represents a sample and each column represents a measurement (or a *feature*) of the sample. Graphs are among a group of objects called semistructured data that cannot easily conform to a matrix representation. Other examples in the group include sequences, cycles, and trees. Though many different features have been proposed for graph data (e.g., paths, cycles, and subgraphs), there is no universally accepted way to define features for graph data.

Besides choosing the right feature representation, computational efficiency is also a serious concern in analyzing graph data. Many graph-related operations, such as subgraph matching, clique identification, and hamiltonian cycle discovery are NP-hard problems. For those that are not NP-hard problems, e.g., all-by-all shortest distance, the computational cost could be prohibitive for large graphs.

In this paper, we aim to leverage existing frequent pattern mining algorithms and explore the application of kernel classifiers in building highly accurate graph classification algorithms. Toward that end, we demonstrate a novel technique called graph pattern diffusion (GPD) kernel. In our method, we first identify all frequent patterns from a graph database. We then

label graph nodes with features denoting membership in frequent patterns, and project nodes of graphs to a high-dimensional space with a specially designed function. Finally, we designed a graph alignment algorithm to compute the inner product of two graphs. We have tested our algorithm using a number of chemical structure data sets. The experimental results demonstrate that our method is significantly better than competing methods such as those based on paths, cycles, and other subgraphs.

In summary, we present the following contributions in this paper:

- A novel way to measure graph similarity using graph kernel functions.
- We prove that the exact computation of the kernel function is an NP-hard problem and we have designed an efficient algorithm to approximately compute the graph kernel function.
- We have implemented our kernel function and tested it with a series of cheminformatics data sets. Our experimental study demonstrates that our algorithm performs much better than existing state-of-the-art graph classification algorithms.

The rest of the paper is organized as follows: In Section 2, we discuss the research efforts that are closely related to our current effort. In Section 3, we define important concepts such as labeled graphs and graph kernel function, and clearly layout the graph classification problem. In Section 4, we present the details of our way of measuring graph similarity with kernel functions. In Section 5, we use real-world data sets to evaluate our proposed methods and perform a comparison of ours to the current state of the art. Finally, we conclude and present our future plan in Section 6.

2 RELATED WORK

We survey the work related to graph classification methods by dividing them into two categories. The first category of methods explicitly collect a set of *features* from the graphs. Possible choices are paths, cycles, trees, and general subgraphs [45]. Once a set of features is determined, a graph is described by a feature vector, and any existing classification methods such as Classification Based on Association (CBA) [4] and decision tree [28] that work in an *n*-dimensional euclidian space, may be applied for graph classification.

The second approach is to implicitly collect a (possibly infinite) set of features from graphs. Rather than computing the features, this approach computes the similarity of graphs, using the framework of “kernel functions” [37].

In what follows, we first give a brief review of pattern discovery algorithms from graphs. Those algorithms provide features for graph classification. We then review the first category algorithms, which explicitly utilize identified features. We delay the discussion of graph kernel functions to Section 3 where we discuss kernel function in general and graph kernel functions specifically.

2.1 Pattern Discovery

Many of the graph pattern mining algorithms adopt a similar criterion to select patterns in a graph database. Given a graph *G*, its *support value* in a graph database is the number of times the graph occurs in the database.[1] If this number is sufficiently large (as compared to a user specified threshold), the graph *G* is called a *frequent pattern*.

Algorithms that search for frequent patterns can be roughly divided into three categories.

[1]The term *occur* is formally defined using subgraph isomorphic relation, see Section 3 for details.

The first category uses a level-wise search strategy, including A-priori-based graph mining (AGM) [18] and frequent subgraph discovery (FSG) [26]. Algorithms in this category mine frequent patterns in a graph database, starting from selecting frequent single node. From frequent single node, all candidate frequent single edge are proposed and their frequency are determined by a linear scan of the related graph database. Both AGM and FSG develop a graph "join" operation, which takes a pair of graphs with k-edges ($k > 0$) as input and produces the common supergraphs of the two graphs with $(k + 1)$-edges as output. Using the join operation, the algorithms propose candidates and then select frequent ones from the candidate subgraphs.

The second category takes a depth-first search strategy, including gSpan [43] and FFSM [19]. Different from level-wise search algorithms AGM and FSG, depth-first search strategy utilizes a backtrack algorithm to mine frequent subgraphs. Starting from one frequent graph G, depth-first search algorithms enumerate the supergraphs of G, select frequent ones, and perform the search recursively. If none of the supergraphs is frequent, depth-first algorithms backtrack. The advantage of a depth-first search is a better memory utilization since depth-first search keeps one frequent subgraph in memory and enumerate its super-graphs, in contrast to keeping all k-edge frequent subgraph in memory.

The third category of frequent subgraph mining algorithms does not directly work on a graph space to identify frequent ones. Algorithms in this category first project a graph space to another space such as that of trees, then identify frequent patterns in the projected space, and finally reconstruct all frequent patterns in the graph space. We call this strategy *progressive mining*. Algorithms in this category include SPIN [15] and GASTON [27].

2.2 Graph Classification

In graph classification, each graph is associated with a target value and the task is to estimate a good function that maps graphs to their target values. The existing algorithms of classifying graph data can be divided into two categories. The first approach is to explicitly collect a set of *features* from the graphs. Possible choices are paths, cycles, trees, and subgraphs. Once a set of features is determined, a graph is described by a feature vector. With a collection of vectorized graph data, any existing data mining method that works in n-dimensional euclidian space may be applied to do graph classification. In the context of cheminformatics, explicit pattern features for compounds are often known as *structural keys*. A structural key is a bit string denoting presence of certain patterns (such as paths, cycles, trees, etc.) of interest. This approach has some similarities to our method, in that they both use pattern information to label graph data. However, in the case of structural keys and other approaches in this category, the patterns label a graph as a whole, while in our method the patterns label individual graph nodes.

The second approach of graph classification is to *implicitly* collect a set of features (possibly an infinite number of such features) and compute the similarity of two graphs via a kernel function. The term *kernel function* refers to an operation of computing the inner product between two points in a Hilbert space, thus may avoid the explicit computation of coordinates in that feature space. *Graph kernel* functions are simply kernel functions that have been defined to compute the inner product between two graphs. In recent years, a variety of graph kernel functions have been developed, with promising application results as described by Ralaivola et al. [29]. Among these methods, some kernel functions draw on graph features such as walks [22] or cycles [12], while others may use different approaches such as genetic algorithms [3], frequent subgraphs [8], or graph alignment [9]. Below, we review some graph kernels.

Kashima et al. [22] proposed a kernel function called the marginalized graph kernel. This kernel function is based on the use of shared label sequences in the comparison of graphs. Their marginalized graph kernel uses a Markov model to randomly generate walks of a labeled graph, based on a transition probability matrix combined with a walk termination probability. These collections of random walks are then compared and the number of shared sequences is used to determine the overall similarity between two graphs.

The optimal-assignment (OA) kernel, proposed by Fröhlich et al. [9], differs significantly from the marginalized graph kernel in that it attempts to align two graphs, rather than compare sets of linear substructures. This kernel function first computes the similarity between all vertices in one graph and those in another. The similarity between the two graphs is then computed by finding the maximal weighted bipartite graph between the two sets of vertices, called the *optimal assignment*. The authors investigate an extension of this method whereby certain structure patterns defined a priori by expert knowledge, are collapsed into single vertices, and this reduced graph is used as input to the optimal-assignment kernel. One drawback to the optimal-assignment kernel, is that it was recently proven not positive definite [38], and hence not a Mercer kernel.

The use of a non-Mercer kernel, while technically not guaranteeing a positive semidefinite kernel matrix, can in practice be a viable approach. Computing a kernel by finding a maximal weighted bipartite matching was shown to be practically useful by Fröhlich et al. [9]. The Fröhlich kernel was later shown to be a non-Mercer kernel, and hence the classifier may not find an optimal solution, yet it still performs well in practice. There has also been work on probabilistic kernels which are positive semidefinite with a high probability [5]. The kernel function proposed here is based on the optimal-assignment kernel and so is also not positive definite. The kernel classifier may not be able to converge to an optimal solution in all cases, but given our results it is able to find reasonable solutions competitive with true Mercer kernels. In practice, many of the eigenvalues of the kernel matrix obtained by our method are close to zero (the smallest differ by less than 10^{-10}), with a some larger positive eigenvalues.

Two relevant nonkernel methods for graph classification are Xrules and graph boosting. XRules [45] utilizes frequent tree patterns to build a rule-based classifier for XML data. Specifically, XRules first identifies a set of frequent tree patterns. An association rule: $G \rightarrow c_i$ is then formed where G is a tree pattern and c_i is a class label. The *confidence* of the rule is the conditional probability $p(c_i \mid G)$ estimated from the training data. XRules carefully selects a subset of rules with high confidence values and uses those rules for classification.

Graph boosting [25] also utilizes substructures toward graph classification. Similar to XRules, graph boosting uses rules with the format of $G \rightarrow c_i$. Different from XRules, it uses the boosting technique to assign weights to different rules. The final classification result is computed as the weighted majority.

Another relevant kernel function is the diffusion kernel [24]. This graph kernel, like ours, utilizes the idea of letting vertex feature values diffuse to neighbors. However, this kernel is defined much differently from ours. It uses a matrix exponentiation form and an equation that describes spread of heat through continuous media. In contrast, our method uses a simple matching kernel for comparing graphs, and the diffusion process is used only for approximate feature labeling. Another important difference is that our method incorporates frequent pattern features for graph matching and comparison.

In the following discussion, we present the necessary background for a formal introduction to the graph classification problem, and introduce a suite of graph kernel functions for graph classification.

3 BACKGROUND

In this section, we discuss a few important definitions for graph database mining: labeled graphs, subgraph isomorphic relation, graph kernel function, and graph classification.

Definition 3.1—*A* labeled graph G is a quadruple G = V E, Σ; λ where V is a set of vertices or nodes and E ⊆ V × V is a set of undirected edges. Σ is a set of (disjoint) vertex and edge labels, and λ : V ∪ E → Σ is a function that assigns labels to vertices and edges. We assume that a total ordering is defined on the labels in Σ.

A *graph database* is a set of labeled graphs.

Definition 3.2—A graph G′ = (V′ E′; Σ′ λ′ is **subgraph isomorphic** to *G* = (V ; E; Σ λ), denoted by *G*′ ⊆ *G*, if there exists a 1–1 mapping f : V′ → V such that

- ∀*v* ∈ V′, λ ′ (*v*) = λ(f(*v*)),
- ∀(*u*, *v*) ∈ E′,(f(*u*), f(*v*)) ∈ E and
- ∀ (*u*, *v*) ∈ E′, λ′ (*u*, *v*) = λ(f(*u*), f(*v*)).

The function *f* is a *subgraph isomorphism* from graph *G*′ to graph *G*. We say *G*′*occurs* in *G* if *G*′ ⊆ *G*. Given a subgraph isomorphism *f*, the image of the domain *V*′ (*f*(*V*′)) is an *embedding* of *G*′ in *G*.

Example 3.1—Fig. 1 shows a graph database of three labeled graphs. The mapping (isomorphism) $q_1 \rightarrow p_3$, $q_2 \rightarrow p_1$, and $q_3 \rightarrow p_2$ demonstrates that graph *Q* is subgraph isomorphic to *P* and hence *Q occurs in P*. Set $\{p_1, p_2, p_3\}$ is an embedding of *Q* in *P*. Similarly, graph *S* occurs in graph *P* but not *Q*.

Problem statement: Given a graph space *G**, a set of *n* graphs sampled from *G** and the related target values *T* ∈ [0,1]of these graphs $D=\{(G_i, T_i,)\}_{i=1}^{n}$, the binary **graph classification problem** is to estimate a function *F : G** → *T* that maps graphs to their target value.

By *classification*, we assume all target values are discrete values, otherwise it is a *regression* problem. Below, we review several algorithms for graph classification that work within a common framework called kernel functions. The term *kernel function* refers to an operation of computing the inner product between two points in a Hilbert space. Kernel functions are widely used in classification of data in a high-dimensional feature space.

3.1 Kernel Functions for Graphs

Kernel functions (and kernel classifiers such as support vector machines) are powerful computational tools to analyze large volumes of graph data [10]. An advantage of kernel functions is their capability to map a set of data to a high-dimensional Hilbert space without explicitly computing the coordinates of the data. This is done through a special function called *kernel* function.

A binary function *K : X* × *X* → IR is a *positive semidefinite* function if

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

for any $m \in \mathbb{N}$, any selection of samples $x_i \in X$ ($i = [1, n]$), and any set of coefficients $c_i \in$ IR ($i = [1; n]$). A positive semidefinite function ensures the existence of a Hillbert space $\mathcal{H}$ and a map $\Phi : X \rightarrow \mathcal{H}$ such that

$$k(x, x') = \left\langle \Phi(x), \Phi(x') \right\rangle, \tag{2}$$

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 positive semidefinite function is also known as a Mercer kernel function [30], or *kernel* function for simplicity.

By embedding the data space to a Hilbert space, kernel functions provide a uniformed analyzing environment for various data types including graphs, regardless the fact that the original data space may not look like a vector space at all. This strategy is known as the "kernel trick" and it has been applied to various data analysis tasks including classification [37], regression [7], and feature extraction through principal component analysis [31] among others.

Graph kernel functions are kernel functions that have been defined to compute the inner product between two graphs. In recent years, a variety of graph kernel functions have been developed, with promising application results as described by Ralaivola et al. [29]. Among these methods, some kernel functions are computed using graph features such as walks [22] or cycles [12], while others may use different approaches such as genetic algorithms [3], frequent subgraphs [8], or optimal graph alignment [9], although the latter is not guaranteed positive definite and therefore does not compute a Mercer kernel.

4 GRAPH ALIGNMENT KERNELS

Here, we present our design of a pattern diffusion kernel. We start the section by first presenting a general framework. We prove, through a reduction to the subgraph isomorphism problem, that the computational cost of the general framework can be prohibitive for large graphs. We then present our pattern-based graph alignment kernel. Finally, we show a technique we call "pattern diffusion" that can significantly improve graph classification accuracy in practice.

4.1 Graph Similarity Measurement with Alignment

An *alignment* of two graphs G and G' (assuming $|V[G]| \leq |V[G']|$) is a mapping $\pi : V[G] \rightarrow V[G']$. Given an alignment π, we define the similarity between two graphs, as measured by a kernel function k_A, below:

$$k_A(G, G') = \max_{\pi} \sum_{v} k_n(v, \pi(v)) + \sum_{u,v} k_e((u, v), (\pi(u), \pi(v))). \tag{3}$$

Note that this mapping is not strictly 1–1 since the larger graph contains nodes that are not mapped to any node in the smaller graph. However, it is 1–1 if considering only the subset of nodes from the larger graph that are mapped to nodes in the smaller one. The function k_n is a kernel function to measure the similarity of node labels and the function k_e is a kernel function to measure the similarity of edge labels. Equation (3) uses an additive model to compute the similarity between two graphs. The maximal similarity among all possible mappings is defined as the similarity between two graphs.

4.2 NP-Hardness of Graph Alignment Kernel Function

It is no surprise that computing the graph alignment kernel is an NP-hard problem. We prove this with a reduction from the graph alignment kernel to the subgraph isomorphism problem. In the following paragraphs, we assume we have an efficient solver of the graph alignment kernel problem, we show that the same solver can be used to solve the subgraph isomorphism problem efficiently. Since the subgraph isomorphism problem is an NP-hard problem, with the reduction we mentioned before we prove that the graph alignment kernel problem is therefore an NP-hard problem as well. Note: this section is a stand-alone component of our paper, and readers who choose to skip this section should encounter no difficulty in reading the rest of the paper.

Given two graphs G and G' (for simplicity, assume nodes and edges in G and G' are not labeled as usually studied in the subgraph isomorphism problem), we use a node kernel function that returns a constant 0. We define an edge kernel function $k_e : V[G] \times V[G] \times V[G'] \times V[G'] \rightarrow \mathrm{IR}$ as

$$k_e((u,v),(u',v')) = \begin{cases} 1, & \text{if } (u,v) \in E[G] \text{ and } (u',v') \in E[G'], \\ 0, & \text{othrewise.} \end{cases}$$

With the constant node function and the specialized edge function, the kernel function of two graphs is simplified to the following format:

$$k_A(G,G') = \max_{\pi} \sum_{u,v} k_e((u,v),(\pi(u),\pi(v))). \tag{4}$$

We establish the NP-hardness of the graph alignment kernel with the following theorem:

Theorem 4.1—Given two (unlabeled) graphs G and G′ and the edge kernel function k_e defined previously, G is subgraph isomorphic to G′ if and only if K_a (G, G′) = |E[G]|.

Proof—If: We notice from the definition of k_e that the maximal value of $K_a(G; G')$ is |$E[G]$|. Given $K_a(G; G') = |E[G]|$, we claim that there exists an alignment function $\pi : V[G] \rightarrow V[G']$ such that for all $(u, v) \in E[G]$ we have $(\pi(u),\pi(v)) \in E[G']$. The existence of such a function π guarantees that graph G is a subgraph of G'.

Only if: Given G is a subgraph of G', we have an alignment function $\pi : V[G] \rightarrow V[G']$ such that for all $(u, v) \in E[G]$ we have $(\pi(u),\pi(v)) \in E[G']$. According to (4), $K_a(G; G') = |E[G]|$.

Theorem 4.1 shows that the graph alignment kernel problem is no easier than the subgraph isomorphism problem and hence is at least NP-hard in complexity.

4.3 Graph Node Alignment Kernel

To derive an efficient algorithm scalable to large graphs, our idea is that we use a function f to map nodes in a graph to a high (possibly infinite) dimensional feature space that captures not only the node label information but also the neighborhood topological information around the node. If we have such function f, we may simplify the graph kernel function with the following formula:

$$k_M(G, G') = \max_{\pi} \sum_{v \in V[G]} k_n(f(v), f(\pi(v))), \tag{5}$$

where $\pi : V[G] \rightarrow V[G']$ denotes an alignment of graph G and G'. $f(v)$ is a set of "features" associated with a node.

With this modification, the optimization problem that searches for the best alignment can be solved in polynomial time. To derive a polynomial running time algorithm, we construct a weighted complete bipartite graph by making every node pair (u,v) $\in V[G] \times V[G']$ incident on an edge. The weight of the edge (u,v) is $k_n f(v), f(u)$. In Fig. 2, we show a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$.

With the bipartite graph, a search for the best alignment becomes a search for the maximum weighted bipartite subgraph from the complete bipartite graph. Many network-flow-based algorithms (e.g., linear programming) can be used to obtain the maximum weighted bipartite subgraph. We use the Hungarian algorithm with complexity $O(|V[G]|^3)$. For details of the Hungarian algorithm see [1].

Applying the Hungarian algorithm to graph alignment was first explored by Fröhlich et al. [9] for chemical compound classification. In contrast to their algorithm, which utilized domain knowledge of chemical compounds extensively and developed a complicated recursive function to compute the similarity between nodes, we develop a new framework that maps such nodes to a high-dimensional space in order to measure the similarity between two nodes without assuming any domain knowledge. Even in cheminformatics, our experiments show that our technique generate similar and sometimes better classification accuracies compared to the method reported in [9].

Unfortunately, using the Hungarian algorithm for assignment, as used by Fröhlich et al. [9] does not give rise to a true Mercer kernel [38]. Since our proposed kernel function uses this algorithm as well, it is also not a Mercer kernel. Like in [9], however, we have found that practically our kernel still performs competitively.

4.4 Pattern Features

The patterns mined through the use of frequent subgraph mining are used as features to label graph nodes when calculating the kernel between two graphs. Given a set of frequent subgraphs S, each node is labeled with a binary feature vector of length $|S|$. Each bit in the vector corresponds to a frequent pattern. If a graph node has the ith bit set to one, then it is part of the ith subgraph pattern, otherwise a zero denotes nonmembership. These binary features are first computed and then a diffusion process, described in the next section, is used to induce approximate matching of these subgraph pattern features.

4.5 Pattern Diffusion

In this section, we introduce a novel function "pattern diffusion" to project nodes in a graph to a high-dimensional space that captures both node labeling information and local topology information. Our design has the following advantages as a kernel function:

- Our design is generic and does not assume any domain knowledge from a specific application. The diffusion process may be applied to graphs with dramatically different characteristics.
- The diffusion process is straightforward to implement and can be computed efficiently.

- We prove that the diffusion process is related to the probability distribution of a graph random walk (in Appendix). This explains why the simple process may be used to summarize local topological information.

Below, we outline the pattern diffusion kernel in three steps.

In the first step, we identify a seed as a starting point for the diffusion. In our design, a "seed" could be a single node, or a set of connected nodes in the original graph. In our experimental study, we use frequent subgraphs for seeds since we can easily compare a seed from one graph to a seed in another graph. However, there is no requirement that we must use frequent subgraphs.

In the second step, given a set of nodes S as seed, we recursively define f_t in the following way.

The base f_0 is defined as:

$$f_0(u) = \begin{cases} 1/|S|, & \text{if } u \in S, \\ 0, & \text{otherwise.} \end{cases}$$

Given some time t, we define f_{t+1} ($t \geq 0$) with f_t in the following way:

$$f_{t+1}(v) = f_t(v) \times \left(1 - \frac{\lambda}{d(v)}\right) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)}. \tag{6}$$

In the notation, $N(v)$ is the set of nodes that connects to v directly. $d(v)$ is the node degree of v, or $d(v) = |N(v)|$. λ is a parameter that controls the diffusion rate.

Equation (6) describes a process where each node distributes a λ fraction of its value to its neighbors evenly and in the same way receives some value from its neighbors. We call it "diffusion" because the process simulate the way a value is spreading in a network. Our intuition is that the distribution of such a value encodes information about the local topology of the network.

To constrain the diffusion process to a local region, we use one parameter called diffusion time, denoted by τ to control the diffusion process. Specifically, we limit the diffusion process to a local region of the original graph with nodes that are at most τ hops away from a node in the seed S. For this reason, the diffusion is referred to as "local diffusion."

Finally, for the seed S, we define the mapping function f_S as the limit function of f_t as t approaches to infinity, or

$$f_S = \lim_{t \to \infty} f_t. \tag{7}$$

4.6 Pattern Diffusion Kernel and Graph Classification

In this section, we summarize the discussion of kernel function and show how the kernel function is utilized to construct an efficient graph classification algorithm at both the training and testing phases.

4.6.1 Training Phase—In the training phase, we divide graphs of the training data set $D=\{(G_i, T_i,)\}_{i=1}^{n}$ into groups according to their class labels. For example, in binary classification, we have two groups of graphs: positive or negative. For multiclass classification, we have multiple groups of graphs where each group contains graphs with the same class label. The training phase is composed of four steps:

- Obtain frequent subgraphs for seeds. We identify frequent subgraphs from each graph group and union the subgraph sets together as our seed set S.
- For each seed $s \in S$ and for each graph G in the training data set, we use f_s to label nodes in G. Thus, the feature vector of a node v is a vector $L_v=\{f_{s_i}(v)\}_{i=1}^{m}$ with length $m = |S|$.

  For two graphs $G; G'$, we construct the complete weighted bipartite graph as described in Section 4.3 and compute the kernel $K_a(G; G')$ using (5).

  Train a predictive model using a kernel classifier.

4.6.2 Testing Phase—In the testing phase, we compute the kernel function for graphs in the testing and training data sets. We use the trained model to make predictions about graph in the testing set.

- For each seed s 2 S and for each graph G in the testing data set, we use f_s to label nodes in G and create feature vectors as we did in the training phase.
- We use (5) to compute the kernel function $K_a(G; G')$ for each graph G in the testing data set and for each graph G' in the training data set.
- Use kernel classifier and trained models to obtain prediction accuracy of the testing data set.

Below we present our empirical study of different kernel functions including our pattern diffusion kernel.

5 EXPERIMENTAL STUDY

We have conducted classification experiments using 10 different biological activity data sets, and compared cross-validation accuracies for different kernel functions. In each cross-validation fold, the data set is divided into training and testing segments. Another internal cross-validation experiment is performed on the training data in order to select the proper SVM model parameters. Then, the model is trained with the training data and validated using the test data. In the following sections, we describe the data sets and the classification methods in more detail along with the associated results.

We performed all of our experiments on a desktop computer with a 3 GHz Pentium 4 processor and 1 GB of RAM. Generating a set of frequent subgraphs is efficient, generally taking a few seconds. Computing alignment kernels somewhat takes more computation time, typically in the range of a few minutes.

In all kernel classification experiments, we used the LibSVM classifier [6] as our kernel classifier. We used v-SVC and selected v from the range 0.1–0.5 using a series of cross-validation experiments, in increments of 0.1; v > 0:5 was not used because the SVM problem became infeasible for many data sets. Our classification accuracy is computed by averaging over 10 additional trials of a 10-fold cross-validation experiment where the proper v has been selected through cross validation on the training data. P-values are calculated using ANOVA test. Several experimental parameters are summarized in Table 2.

5.1 Data Sets

We have selected 10 data sets covering typical chemical benchmarks in drug design to evaluate our classification algorithm performance.

The first five data sets are from drug virtual screening experiments taken from [21]. In this data set, the target values are drugs' binding affinity to a particular protein. Five proteins are used in the data set including: CDK2, COX2, FXa, PDE5, and A1A where each symbol represents a specific protein. For each protein, the data provider carefully selected 50 chemical structures that clearly bind to the protein ("active" ones). The data provider also deliberately 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. Since our goal is to evaluate classifiers, we will not further elaborate the nature of the data set. See [21] for details.

The next data set, from Wessel et al. [40] includes compounds classified by affinity for absorption through human intestinal lining. Moreover, we included the Predictive Toxicology Challenge [11] data sets, which contain a series of chemical compounds classified according to their toxicity in male rats, female rats, male mice, and female mice.

We use the same way as was done in [14] to transform chemical structure data set to graphs. In Table 1 for each data set, we list the total number of chemical compounds in the data set, as well as the number of positive and negative samples.

5.2 Feature Sets

We used frequent patterns from graph represented chemicals exclusively in our study. We generate such frequent subgraphs from a data set using two different graph mining approaches: that with exact matching [14] and that of approximate matching. In our approximate frequent subgraph mining, we consider that a pattern *matches* with a graph as long as there are up to $k > 0$ node label mismatches. For chemical structures typical mismatch tolerance is small, that is k values are 1, 2, etc. In our experiments, we used approximate graph mining with $k = 1$.

Once frequent subgraphs are mined, we generate three feature sets: 1) general subgraphs (all of mined subgraphs), 2) tree subgraphs, and 3) path subgraphs. We tried cycles as well, but did not include them in this study since typically less than two cyclic subgraphs were identified in a data set. These feature sets are used for constructing kernel functions as discussed below.

5.3 Classification Methods

We have evaluated the performance of the following classifiers:

- CBA. The first is a classifier that uses frequent item set mining, known as CBA [4]. In CBA, we treat mined frequent subgraphs as item sets.
- Graph Convolution Kernels. This type of kernel includes the mismatch kernel (MIS), based on the normalized Hamming distance of two binary vectors.
- SVM built-in Kernels. We used linear kernel (Linear) and radial basis function (RBF) kernel.
- GPD. We implemented the graph pattern diffusion kernel as discussed in Section 4. The parameters for the GPD kernel are diffusion rate $\lambda = 20$ percent and diffusion time $\tau = 5$.

5.4 Experimental Results

Here, we present the results of our graph classification experiments. We perform one round of experiments to evaluate the methods based on exact subgraph mining, and another round of experiments with approximate subgraph mining. For both of these two subgraph mining methods, we selected patterns that were general graphs, tree graphs, and cycles.

We perform feature selection in order to identify the most discriminating frequent patterns. This supervised process is performed using the training data from each cross-validation trial. Using a simple statistical formula, Pearson correlation coefficient (PCC), we measure the correlation between a set of feature samples (in our case, the occurrences of a particular subgraph in each of the data samples) and the corresponding class labels. Frequent patterns are ranked according to correlation strength, and the top 10 percent patterns are selected to construct the feature set.

5.4.1 Comparison between Classifiers—The results of the comparison of different graph kernel functions are shown in Tables 4 and 5, along with p-values for the significance of GPD method compared to others. From the tables, we observe that our GPD method outperforms the other methods on many of the data sets, particularly the protein target data. While the linear kernel method performs best for to toxicity data sets. In many of these cases, the difference is significant according to p-values from an ANOVA test. The GPD's performance is also confirmed in classifications where tree and path patterns are used.

In Table 3, we compare the performance of our GPD kernel to the CBA method, or Classification Based on Association. In general, it shows comparable performance to the other methods. In one data set, it does show a noticeable increase over the other methods. This is expected since CBA is designed specifically for discrete data such as the binary feature occurrences used here. Despite the strengths of CBA, we can see that the GPD method still gives the best performance for six of the seven data sets. The results for the other toxicology data sets are not shown here because CBA generated errors for this data. We also tested these data sets using the recursive OA kernel included in the JOELib2 computational chemistry library, and compare this method to CBA and GPD. This method uses optimal assignment but not pattern features. The features used instead are atom element and bond type. Note that for the FXa data set, there is no result for OA since it gives an error on this data set.

In addition, we tested a classifier called XRules. XRules is designed for classification of tree data [45]. Chemical graphs, while not strictly trees, often are close to trees. To run the XRules executable, we transform a graph to a tree by randomly selecting a spanning tree of the original graph. The results show that the application of XRules on average delivers decreased performance results among the group of classifiers (e.g., 50 percent accuracy on the CDK2 inhibitor data set), which may be due to the particular way we transform a graph to a tree. Since we compute tree patterns for rule-based classifier such as CBA in our comparison, we did not explore further of XRules.

We also tested a method based on a recursive OA [9] using biologically relevant chemical descriptors labeling each node in a chemical graph. In order to perform a fair comparison with this method to the other methods, we chose to ignore the chemical descriptors and focus on the structural alignment.

5.4.2 Comparison between Descriptor Sets—Various types of subgraphs such as trees, paths, and cycles have been used in kernel functions between chemical compounds. In addition to exact mining of general subgraphs, we also chose to use approximate subgraph mining to generate the features for our respective kernel methods. In both cases, we filtered

the general subgraphs mined into sets of trees and sets of paths as well. The results for these experiments are given in Tables 4 and 5. The results for approximate patterns are not reported for toxicity data sets due to errors when processing this data.

From Table 4, we see that the results for all kernels using exact tree subgraphs are identical to those for exact general subgraphs. This is not surprising, given that most chemical fragments are structured as trees. The results using exact path subgraphs, however, do show some shifts in accuracy but the difference is not significant.

The results using approximate subgraph mining (shown in Table 5) in contrast to those for exact subgraph mining (shown in Table 4). The use of approximate patterns appears to increase the classification accuracy of most other methods more GPD, this makes some amount of sense considering GPD already uses a diffusion process to approximate the features.

5.4.3 Effect of Varying GPD Diffusion Rate and Time—We want to evaluate the sensitivity of the GPD methods to its two parameters: diffusion rate λ and diffusion time. We tested different diffusion rate λ values and diffusion time values. Each parameter was varied while the other was held constant.

We can see from Fig. 3 that diffusion rate has various effects on the different data sets. In some cases, more diffusion is desirable and in other cases less. We do note that some of the data sets benefit a great deal from diffusion; in particular, the intestinal absorption data show an increase of 8 percent from using diffusion.

From Fig. 3, we can also see that the performance differences when varying diffusion time follow the same pattern as those for diffusion rate. That is, most data sets respond marginally to changes in the parameter, with at least some diffusion preferred. At the longer diffusion times, the results for some data sets degrade, while for other data sets they improve.

From these data, we can observe the performance of the GPD method under three different diffusion conditions: no diffusion, some diffusion, and convergent diffusion. The results indicating no diffusion (where diffusion rate and time are equal to zero) correspond to exact matching. Convergence arises when the diffusion process is performed until the approximate labels stop changing. This situation can be seen in the cases where diffusion is carried out quickly (with high diffusion rate) or for a long time (long diffusion time). In between these two cases is the situation where some diffusion has occurred but not until convergence, which can be seen in the experiments with median parameter values.

Different data sets respond differently to parameters and level of diffusion. Some data such as intestinal absorption seem to favor a higher degree of diffusion, while others favor less, such as the toxicity data sets. Still others seem insensitive to diffusion.

6 CONCLUSIONS AND FUTURE WORKS

With the rapid development of fast and sophisticated data collection methods, data have become complex, high dimensional, and noisy. Graphs have proven to be powerful tools for modeling complex, high-dimensional, and noisy data; building highly accurate predictive models for graph data is a new challenge for the data mining community. In this paper, we have demonstrated the utility of a novel graph kernel function, GPD kernel. We showed that the GPD kernel can capture the intrinsic similarity between two graphs and has the lowest testing error in many of the data sets evaluated. Although we have developed a very efficient computational framework, computing a GPD kernel may be hard for large graphs. Our

future work will concentrate on improving the computational efficiency of the GPD kernel for very large graphs, as well as performing additional comparisons between our method other 2D-descriptor and QSAR-based methods.

Acknowledgments

This work has been supported by the Kansas IDeA Network for Biomedical Research Excellence (NIH/NCRR award #P20 RR016475) and the KU Center of Excellence for Chemical Methodology and Library Development (NIH/ NIGM award #P50 GM069663).

References

1. Ahuja R, Magnanti T, Orlin J. Network Flows. SIAM Rev. 1995;37(1)
2. Austin C, Brady L, Insel T, Collins F. Nih Molecular Libraries Initiative. Science. 2004;306(5699): 1138–1139. [PubMed: 15542455]
3. Barbu, E.; Raveaux, R.; Locteau, H.; Adam, S.; Heroux, P. Graph Classification Using Genetic Algorithm and Graph Probing Application to Symbol Recognition. Proc. 18th Int'l Conf. Pattern Recognition (ICPR); 2006.
4. Bing Liu, YM.; Hsu, W. Integrating Classification and Association Rule Mining. Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining; 1998.
5. Boughorbel, S.; Tarel, J.; Fleuret, F. Non-Mercer Kernels for SVM Object Recognition. Proc. British Machine Vision Conf.,; 2004.
6. Chang, C.; Lin, C. Libsvm: A Library for Support Vector Machines. 2001. http://www.csie.ntu.edu.tw/cjlin/libsvm
7. Collobert R, Bengio S. SVMTorch: Support Vector Machines for Large-Scale Regression Problems. J Machine Learning Research. 2001;1:143–160.
8. Deshpande M, Kuramochi M, Karypis G. Frequent SubStructure-Based Approaches for Classifying Chemical Compounds. IEEE Trans Knowledge and Data Eng. Aug; 2005 17(8):1036–1050.
9. Fröhlich, Wegner J, Sieker F, Zell A. Kernel Functions for Attributed Molecular Graphs—A New Similarity-Based Approach to ADME Prediction in Classification. QSAR & Combinatorial Science. 2006;25:317–326.
10. Haussler, D. Technical Report UCSC-CRL099-10. Computer Science Dept; UC Santa Cruz: 1999. Convolution Kernels on Discrete Structures.
11. Helma C, King R, Kramer S. The Predictive Toxicology Challenge 2000–2001. Bioinformatics. 2001;17(1):107–108.
12. Horvath, T.; Gartner, T.; Wrobel, S. Cyclic Pattern Kernels for Predictive Graph Mining. Proc. ACM SIGKDD; 2004.
13. Horvath, T.; Ramon, J.; Wrobel, S. Frequent Subgraph Mining in Outerplanar Graphs. Proc. ACM SIGKDD; 2006.
14. Huan, J.; Wang, W.; Prins, J. Efficient Mining of Frequent Subgraph in the Presence of Isomorphism. Proc. Third IEEE Int'l Conf. Data Mining (ICDM); 2003. p. 549-552.
15. Huan, J.; Wang, W.; Prins, J.; Yang, J. SPIN: Mining Maximal Frequent Subgraphs from Graph Databases. Proc. ACM SIGKDD; 2004. p. 581-586.
16. Huan, J.; Wang, W.; Washington, A.; Prins, J.; Shah, R.; Tropsha, A. Accurate Classification of Protein Structural Families Based on Coherent Subgraph Analysis. Proc. Pacific Symp. Biocomputing (PSB),; 2004. p. 411-422.
17. Huang Y, Li H, Hu H, Yan X, Waterman MS, Huang H, Zhou XJ. Systematic Discovery of Functional Modules and Context-Specific Functional Annotation of Human Genome. Bioinformatics. 2007;23:222–229. [PubMed: 17110366]
18. Inokuchi, A.; Washio, T.; Motoda, H. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. Proc. Conf. Principles of Data Mining and Knowledge Discovery (PKDD '00); 2000. p. 13-23.
19. Huan, WWJ.; Prins, J. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. Proc. IEEE Int'l Conf. Data Mining (ICDM); 2003.

20. Jin, R.; Mccalle, S.; Almaas, E. Trend Motif: A Graph Mining Approach for Analysis of Dynamic Complex Networks. Proc. IEEE Int'l Conf. Data Mining (ICDM); 2007.
21. Jorissen R, Gilson M. Virtual Screening of Molecular Databases Using a Support Vector Machine. J Chemical Information and Modeling. 2005;45(3):549–561.
22. Kashima, H.; Tsuda, K.; Inokuchi, A. Marginalized Kernels between Labeled Graphs. Proc. 20th Int'l Conf. Machine Learning (ICML); 2003.
23. Ke, Y.; Cheng, J.; Ng, W. Correlation Search in Graph Databases. Proc. ACM SIGKDD; 2007.
24. Kondor, R.; Lafferty, J. Diffusion Kernels on Graphs and Other Discrete Input Spaces. Proc. Int'l Conf. Machine Learning (ICML); 2002.
25. Kudo, T.; Maeda, E.; Matsumoto, Y. An Application of Boosting to Graph Classification. Proc. Neural Information Processing Systems (NIPS) Conf; 2004.
26. Kuramochi, M.; Karypis, G. Frequent Subgraph Discovery. Proc. Int'l Conf. Data Mining 2001; 2001. p. 313-320.
27. Nijssen, S.; Kok, J. A Quickstart in Frequent Structure Mining Can Make a Difference. Proc. 10th ACM SIGKDD; 2004. p. 647-652.
28. Quinlan, JR. C4.5: Programs for Machine Learning. Morgan Kaufmann; 1993.
29. Ralaivola L, Swamidass SJ, Saigo H. Graph Kernels for Chemical Informatics. Neural Networks. 2005;18:1093–1110. [PubMed: 16157471]
30. Schölkopf, B.; Smola, AJ. Learning with Kernels. The MIT Press; 2002.
31. Schölkopf, B.; Smola, AJ.; Müller, KR. Advances in Kernel Methods: Support Vector Learning. The MIT Press; 1999. Kernel Principal Component Analysis; p. 327-352.
32. Shasha, D.; Wang, JTL.; Giugno, R. Algorithmics and Applications of Tree and Graph Searching. Proc. ACM Symp. Principles of Database Systems (PODS); 2002.
33. Smalter, A.; Huan, J.; Lushington, G. Structure-Based Pattern Mining for Chemical Compound Classification. Proc. Sixth Asia Pacific Bioinformatics Conf; 2008.
34. Sun, J.; Papadimitriou, S.; Yu, PS.; Faloutsos, C. Parameter-Free Mining of Large Time-Evolving Graphs. Proc. ACM SIGKDD; 2007.
35. Tolliday, N.; Clemons, PA.; Ferraiolo, P.; Koehler, AN.; Lewis, TA.; Li, X.; Schreiber, SL.; Gerhard, DS.; Eliasof, S. Small Molecules, Big Players: The National Cancer Institute's Initiative for Chemical Genetics; Cancer Research; 2006. p. 8935-8942.
36. Tong, H.; Koren, Y.; Faloutsos, C. Fast Direction-Aware Proximity for Graph Mining. Proc. ACM SIGKDD; 2007.
37. Vapnik, V. Statistical Learning Theory. John Wiley; 1998.
38. Vert, P. Computing Research Repository (CoRR), abs/0801.4061. 2008. The Optimal Assignment Kernel is not Positive Definite.
39. Wale N, Watson I, Karypis G. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. Knowledge and Information Systems. 2008;14:347–375.
40. Wessel M, Jurs P, Tolan J, Muskal S. Prediction of Human Intestinal Absorption of Drug Compounds from Molecular Structure. J Chemical Information and Computer Sciences. 1998;38(4):726–735.
41. Weston J, Kuang R, Leslie C, Noble WS. Protein Ranking by Semi-Supervised Network Propagation. BMC Bioinformatics. 2006;7
42. Williams, D.; Huan, J.; Wang, W. Graph Database Indexing Using Structured Graph Decomposition. Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE); 2007.
43. Yan, X.; Han, J. gSpan: Graph-Based Substructure Pattern Mining. Proc. Int'l Conf. Data Mining 2002; 2002. p. 721-724.
44. Yan, X.; Yu, PS.; Han, J. Graph Indexing Based on Discriminative Frequent Structure Analysis. ACM Trans. Database Systems; 2005. p. 960-993.
45. Zaki MJ, Aggarwal CC. Xrules: An Effective Structural Classifier for Xml Data. Machine Learning J. 2006;62(1/2):137–170. special issue on statistical relational learning and multi-relational data mining.
46. Zeng, Z.; Wang, J.; Zhou, L.; Karypis, G. Coherent Closed Quasi-Clique Discovery from Large Dense Graph Databases. Proc. ACM SIGKDD; 2006.

47. Zhou, D.; Huang, J.; Schöolkopf, B. Learning from Labeled and Unlabeled Data on a Directed Graph. Proc. 22nd Int'l Conf. Machine Learning; 2005.

APPENDIX

Here, we show the connection of pattern diffusion kernel function to the marginalized graph kernel [22], which uses a Markov model to randomly generate walks of a labeled graph.

Given a graph G with nodes set $V[G] = \{v_1, v_2, \ldots, v_n\}$, and a seed $S \subseteq V[G]$, for each diffusion function f_t, we construct a vector $U_t = (f_t(v_1), f_t(v_2), \ldots, f_t(v_n))$. According to the definition of f_t, we have $U_{t+1} = \Gamma \times U_t$ where the matrix Γ is defined as:

$$\Gamma(i,j)=\begin{cases} \frac{\lambda}{d(v_j)}, & \text{if } i \neq j \text{ and } i \in N(j), \\ 1-\frac{\lambda}{d(v_i)}, & i=j, \\ 0, & \text{otherwise.} \end{cases}$$

In this representation, we compute the stationary distribution ($f_S = \lim_{t\to\infty} f_t$) by computing $\Gamma^{\infty} \times U_0$.

We notice that the matrix Γ corresponds to a probability matrix corresponding to a Markov Chain since

- all entries are non-negative and
- column sum is 1 for each column.

Therefore, the vector $\Gamma^{\infty} \times U_0$ corresponds to the stationary distribution of the local random walk as specified by Γ. In other words, rather than using random walk to retrieve information about the local topology of a graph, we use the stationary distribution to retrieve information about the local topology. Our experimental study shows that this, in fact, is an efficient way for graph classification.

Biographies

Aaron Smalter is a PhD student in computer science in the Department of Electrical Engineering and Computer Science, University of Kansas (KU). He is a graduate research assistant with the Molecular Graphics and Modeling Laboratory and is also affiliated with the Information and Telecommunication Technology Center at KU.

Jun (Luke) Huan received the PhD degree in computer science from the University of North Carolina at Chapel Hill in 2006. He has been an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Kansas (KU) since 2006. He is an affiliated member of the Information and Telecommunication Technology Center (ITTC), Bioinformatics Center, Bioengineering Program, and the Center for Biostatistics and Advanced Informatics—all KU research organizations. Before joining KU, he worked at the Argonne National Laboratory (with Ross Overbeek) and GlaxoSmithKline plc. (with Nicolas Guex).

Yi Jia is a PhD candidate in the Department of Electrical Engineering and Computer Science, University of Kansas (KU). He is also a part of the Intelligent Systems Laboratory and the Bioinformatics and Computation Life-Sciences Laboratory in the Information and Telecommunication Technology Center at KU.

Gerald Lushington received the PhD degree in theoretical chemistry from the University of New Brunswick in 1996. He is currently the director of the Molecular Graphics and Modeling Laboratory at the University of Kansas. He is also a courtesy assistant professor in

the Department of Chemistry and Medicinal Chemistry. He is involved with the K-INBRE Bioinformatics Core facility, directing of a statewide network of informatics cores aimed at providing data analysis and data mining support to academic biological research.

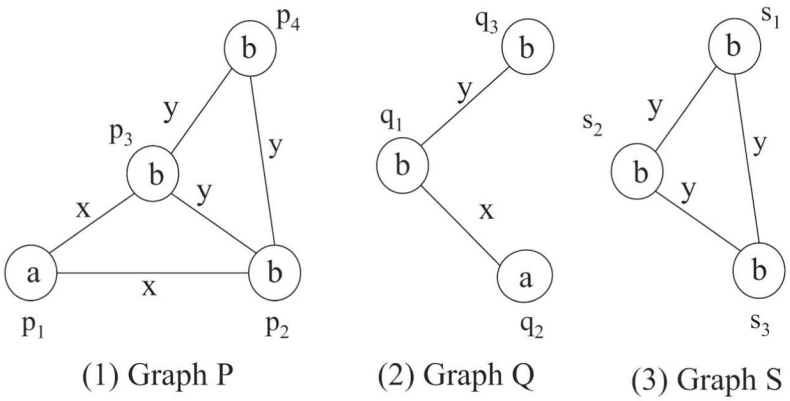

Fig. 1.

A database of three different labeled graphs, each containing some common subgraphs with *Q* and *S* each a subgraph of *P*.

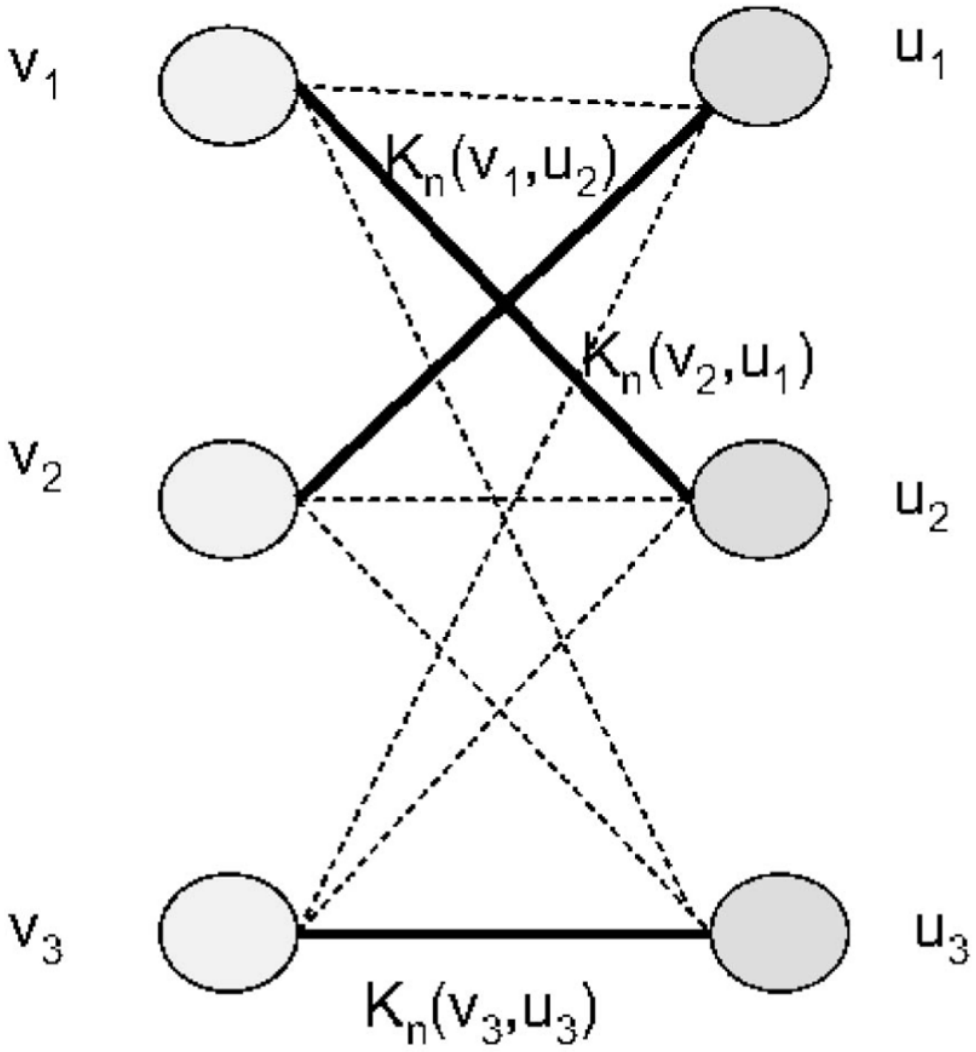

Fig. 2.
The maximum weighted bipartite graph for graph alignment. Highlighted edges (*v*1, *u*2), (*v*2, *u*1), and (*v*3, *u*3) have larger weights than the rest of the edges (dashed).

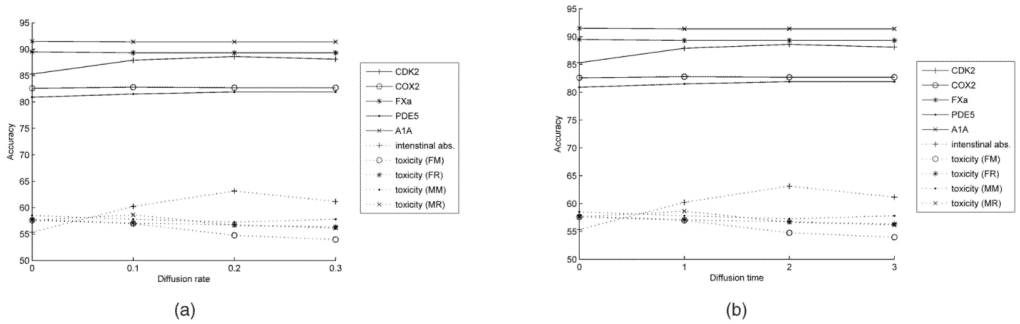

Fig. 3.

(a) GPD Classification Accuracy with different diffusion rate. (b) GPD Classification Accuracy with different diffusion time.

TABLE 1

Data Set and Class Statistics

Dataset	# G	# P	# N
CDK2 inhibitors	100	50	50
COX2 inhibitors	100	50	50
Fxa inhibitors	100	50	50
PDE5 inhibitors	100	50	50
A1A inhibitors	100	50	50
intestinal absorption	310	148	162
toxicity (female mice)	344	152	192
toxicity (female rats)	336	129	207
toxicity (male mice)	351	121	230
toxicity (male rats)	349	143	206

#G: number of samples (chemical compounds) in the data set. # P: positive samples. # N: negative samples.

TABLE 2

Method Parameters Unless Stated Otherwise as in Fig. 3

Method	Parameter	Value/Range
ν-SVC	ν	[0.1,0.2,0.3,0.4,0.5]
Approx. Graph Mining	Mismatch tolerance	1
GPD	λ	0.2
GPD	τ	5
PCC	features selected	%10

Ranged values are chosen by cross validation.

TABLE 3

Comparison of GPD to CBA and Optimal Assignment

Data set	GPD	CBA	OA
CDK2	83*	80	82
COX2	77	77	80*
FXa	88*	86	-
PDE5	89*	87	84
A1A	94*	87	92
intestinal	60*	54	52
toxicity (MR)	51	55*	53

Best accuracy is marked with asterisk.

TABLE 4

Comparison of Different Graph Kernel Functions Using Different Subgraph Feature Minded by FFSM

subgraph type	Data set	GPD	MIS	p-val	Linear	p-val	RBF	p-val
general	CDK2	83*	71.8	0.01	72.9	0.01	74.3	0.02
	COX2	77.6*	54	<0.01	52.6	<0.01	56.7	<0.01
	FXa	88.1*	63.2	<0.01	59.2	<0.01	62.1	<0.01
	PDE5	89.4*	81.7	<0.01	79.5	<0.01	81.9	<0.01
	A1A	94.4*	89.4	<0.01	89.4	<0.01	90.5	<0.01
	intestinal abs.	60.93*	48.95	<0.01	52.09	<0.01	49.77	<0.01
	toxicity (FM)	50.23	53.67	0.08	52.78	0.22	56.93*	0.01
	toxicity (FR)	55.41	54.07	0.46	54.73	0.5	59.97*	<0.01
	toxicity (MM)	54.23	52.23	0.4	49.29	0.07	55.95*	0.53
	toxicity (MR)	51.45	51.45	1	52.7	0.45	59.39*	<0.01
trees	CDK2	83*	71.8	0.01	72.9	0.01	74.3	0.02
	COX2	77.6*	54	<0.01	52.6	<0.01	56.7	<0.01
	FXa	88.1*	63.2	<0.01	59.2	<0.01	62.1	<0.01
	PDE5	89.4*	81.7	<0.01	79.5	<0.01	81.9	<0.01
	A1A	94.4*	89.4	<0.01	89.4	<0.01	90.5	<0.01
	intestinal abs.	60.93*	48.95	<0.01	52.09	<0.01	49.77	<0.01
	toxicity (FM)	50.23	53.67	0.08	52.78	0.22	56.93*	0.01
	toxicity (FR)	55.41	54.07	0.46	54.73	0.5	59.97*	<0.01
	toxicity (MM)	54.23	52.23	0.4	49.29	0.07	55.95*	0.53
	toxicity (MR)	51.45	51.45	1	52.7	0.45	59.39*	<0.01
paths	CDK2	82.2*	63	<0.01	64.5	<0.01	80.2	0.01
	COX2	77.1*	51.2	<0.01	50.6	<0.01	52.7	<0.01
	FXa	89.6*	74.1	<0.01	74.9	<0.01	87.9	0.03
	PDE5	91.3*	73.6	<0.01	70	<0.01	79.7	<0.01
	A1A	93.3*	90.4	<0.01	90.4	<0.01	90.8	<0.01
	intestinal abs.	49.19*	48.14	0.72	48.72	0.87	47.67	0.62
	toxicity (FM)	52.32	54.24	0.41	53.18	0.61	57.25*	0.03

subgraph type	Data set	GPD	MIS	p-val	Linear	p-val	RBF	p-val
	toxicity (FR)	54.44	55.16	0.69	53.93	0.7	59.86*	<0.01
	toxicity (MM)	56.31	51.07	0.04	51.16	0.05	56.46*	0.94
	toxicity (MR)	51.92	52.91	0.51	51.92	1	54.91*	0.05

Best accuracy is marked with asterisk. Accuracy is given as a percent, along with p-values compared to GPD in the second column for other methods.

TABLE 5

Comparison of Different Graph Kernel Functions Using Different Subgraph Features Mined by Approximate Matching

subgraph type	Data set	GPD	MIS	p-val	Linear	p-val	RBF	p-val
general	CDK2	77.4*	58.4	<0.01	58	<0.01	51.9	<0.01
	COX2	80.7	83.2*	0.03	82.3	0.12	80.1	0.71
	FXa	95.7*	95.5	0.72	95.5	0.72	95.3	0.51
	PDE5	88.4*	86.7	0.16	86.7	0.16	85.8	0.03
	A1A	91.3*	56.9	<0.01	50.6	<0.01	54.3	<0.01
	intestinal abs.	59.42*	49.88	<0.01	52.79	<0.01	52.67	0.01
trees	CDK2	77.4*	58.4	<0.01	58	<0.01	51.9	<0.01
	COX2	80.7	83.2*	0.03	82.3	0.12	80.1	0.71
	FXa	95.7*	95.5	0.72	95.5	0.72	95.3	0.51
	PDE5	88.4*	86.7	0.16	86.7	0.16	85.8	0.03
	A1A	91.3*	56.9	<0.01	50.6	<0.01	54.3	<0.01
	intestinal abs.	59.42*	49.88	<0.01	52.79	<0.01	52.67	0.01
paths	CDK2	75.9*	58	<0.01	62.7	0.01	53.1	<0.01
	COX2	81*	78.4	0.08	76.7	0.03	79.1	0.27
	FXa	94.9*	92.7	<0.01	92.6	<0.01	92.5	<0.01
	PDE5	87.7*	78.8	<0.01	79.9	<0.01	86.7	0.16
	A1A	95.8*	70.6	<0.01	72.3	<0.01	83.5	<0.01
	intestinal abs.	53.72*	51.51	0.35	53.14	0.78	50.47	0.19

Best accuracy is marked with asterisk. Accuracy is given as a percent, along with p-values compared to GPD in the second column for other methods.

IEEE/ACM Trans Comput Biol Bioinform. Author manuscript; available in PMC 2011 March 16.