Kernel Functions for Graph Classification

Aaron M. M. Smalter

Submitted to the Department of Electrical Engineering & Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Thesis Committee:

The Thesis Committee for Aaron M. M. Smalter certifies
that this is the approved version of the following thesis:

Kernel Functions for Graph Classification

Committee:

____

Chairperson

____

____

____

____

Date Approved

Acknowledgements

I would like to thank my advisor, Dr. Jun Huan, for his enthusiasm and direction throughout my research work. Truly, to him I owe a great debt for guiding me into this field of work that I enjoy so much. Next, I would like to thank Dr. Gerald Lushington for his support and employ in the Molecular Graphics and Modeling Laboratory, which has been nothing short of invaluable.

I would also like to thank my mother Janis, my father Martin, my brother Colin and the rest of my family, extended and otherwise - without all of you I would never have made it to where I am today. I thank Rachel Diana Hall, for her love and encouragement during these recent months of study, for she has been a beacon throughout many trials. Finally, a hearty thanks to all my friends in Lawrence and everywhere else.

To the University of Kansas I owe not only my education, but also the inspiration I felt as a child in the activities, services, and culture provided to me and everyone else in the Lawrence community. It is these experiences that first showed me the value of learning and knowledge.

This work has been supported by the Kansas IDeA Network for Biomedical Research Excellence (NIH/NCRR award #P20 RR016475), the KU Center of Excellence for Chemical Methodology and Library Development (NIH/NIGM award #P50 GM069663), and NIH grant #R01 GM868665.

Abstract

Graphs are information-rich structures, but their complexity makes them difficult to analyze. Given their broad and powerful representation capacity, the classification of graphs has become an intense area of research. Many established classifiers represent objects with vectors of explicit features. When the number of features grows, however, these vector representations suffer from typical problems of high dimensionality such as overfitting and high computation time. This work instead focuses on using *kernel functions* to map graphs into implicity defined spaces that avoid the difficulties of vector representations.

The introduction of kernel classifiers has kindled great interest in kernel functions for graph data. By using kernels the problem of graph classification changes from finding a good classifier to finding a good kernel function. This work explores several novel uses of kernel functions for graph classification. The first technique is the use of structure based features to add structural information to the kernel function. A strength of this approach is the ability to identify specific structure features that contribute significantly to the classification process. Discriminative structures can then be passed off to domain-specific researchers for additional analysis. The next approach is the use of wavelet functions to represent graph topology as simple real-valued features. This approach achieves order-of-magnitude decreases in kernel computation time by eliminating costly topological comparisons, while retaining competitive classification accuracy. Finally, this work examines the use of even simpler graph representations and their utility for classification. The models produced from the kernel functions presented here yield excellent performance with respect to both efficiency and accuracy, as demonstrated in a variety of experimental studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work of this thesis addresses the problem of graph classification through study of kernel functions. Classification of objects in a vector space is well researched and many methods exist. They are all limited, however, by explicit and often high dimensional feature vectors. Kernel classifiers instead embed objects in an implicit space and hence avoid the difficulties of managing a large number of explicit features. Kernels are especially useful in graph classification where the objects being modeled have a complex structure.

While kernel functions for graphs have received a great deal of attention recently, most approaches are stymied by graph complexity. Precise comparisons are slow to compute, but simpler methods do not capture enough information about graph topology and structure. The focus of this work is to augment simple graph representations with structure information, allowing the use of fast kernel functions while recognizing important topological similarities. This thesis draws from several studies: incorporating structure features graphs into kernel functions [45], extensions for approximate matching of such structure features [42], set-based matching kernels with structure features [43], and an application of wavelets for simplified topology comparison in graph kernels [44].

Graph classification is important for a variety of reasons. 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 [22]. To analyze biological data, graphs have been utilized in modeling chemical structures [45], protein sequences [54], protein structures [16], and gene regulation networks [19]. In web page classification, graphs are used to model the referencing relationship in HTML documents [62].

Of particular importance are chemical activity prediction studies. Here the goal is, given a chemical graph, to predict whether or not it will display some biological activity of interest. The development of accurate models for chemical activity prediction has a range of applications. They are especially useful in the screening of potential drug candidates, currently a difficult and expensive process that can benefit enormously from accurate computational methods. These models have proved difficult to design, due to the complex nature of most biological classification problems. For example, the toxicity of a particular chemical compound is determined by a large variety of factors, as there are innumerable ways that a foreign chemical might interfere with an organism, and the situation is further complicated by the possibility that a benign chemical may be broken down into toxic metabolites in the body. Clearly, there is no single set of chemical features that can be easily applied to to all problems in all situations, and therefore the ability to isolate problem-specific chemical features from broader data collections is a critical issue.

Graph classification for chemical activity prediction must also be computationally efficient. The fast accumulation of data describing chemical compound structures and biological activity calls for the development of efficient informatics tools. *Cheminformatics* is a rapidly emerging research discipline that employs a wide array of statistical, data mining, and machine learning techniques with the goal of establishing robust relationships between chemical structures and their biological properties. Cheminformatics hence is an important component on the application side of applying informatics approach to life science problems. It has a broad range of applications in chemistry and biology; arguably the most commonly known roles are in the area of drug discovery where cheminformatics tools play a central role in the analysis and interpretation of structure-activity data collected by various means of modern high throughput screening technology. Traditionally the analysis of large chemical structure-activity databases was done only within pharmaceutical companies and up until recently the academic community has had only limited access to such databases. This situation, however, has changed dramatically in very recent years.

In 2002, the National Cancer Institute created the Initiative for Chemical Genetics (ICG) with the goal of offering to the academic research community a large database of chemicals with their roles in cancer research [49]. Two years later, the National Health Institute (NIH) launched a Molecular Libraries Initiative (MLI) that included the formation of the national Molecular Library Screening Centers Network (MLSCN). MLSCN is a consortium of 10 high-throughput screening centers for screening large chemical libraries [2]. Collectively, ICG and MLSCN aim to offer to the academic research community the results of testing about a million compounds against hundreds of biological targets. To organize this data and to provide public access to the results, the PubChem database and the Chembank database have been developed as the central repository for chemical structure-activity data. These databases currently contain more than 18 million chemical compound records, more than 1000 bioassay results, and links from chemicals to bioassay description, literature, references, and assay data for each entry.

These publicly-available large-scale chemical compound databases have offered tremendous opportunities for designing highly efficient computational drug design methods.

Many machine learning and data mining algorithms have been applied to study the structure-activity relationship of chemicals. For example, Xue *et al.* reported promising results of applying five different machine learning algorithms: logistic regression, C4.5 decision tree, k-nearest neighbor, probabilistic neural network, and support vector machines to predicting the toxicity of chemicals against an organism of Tetrahymena pyriformis [57]. Advanced techniques, such as random forest and MARS (Multivariate Adaptive Regression Splines) have also been applied to cheminformatics applications [39, 47].

Additionally, development of computational and statistical frameworks for analyzing graph data has attracted significant research attention in the data mining community. In the past a few years, various graph pattern mining algorithms have been designed [15, 17, 46, 50, 58, 61]. There are also many research efforts dedicated to efficiently searching graph databases [25, 41, 55, 59].

It is clear that graph classification is an important area of study. In many existing classification algorithms [3], 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 semi-structured 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 graph features.

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.

Recently Support Vector Machines (SVM) have gained popularity in drug design and cheminformatics. A key insight of SVM is the utilization of kernel functions (i.e. inner product of two points in a Hilbert Space) to transform a non-linear classification problem into a linear one. Design of a good kernel function for graphs is therefore a critical issue. The initial effort to define kernels for semi-structured data was done by Haussler in his work on the *R-convolution* kernel, which provided a framework which many current graph kernel functions follow [13].

Traditional approaches to graph and chemical similarity have a variety of limitations. Methods for chemical activity prediction rely on the comparison of graphs using a variety of molecular attributes known *a priori* to be involved in the activity of interest. Such methods are problem-specific, however, and provide little assistance when the relevant descriptors are not known in advance. Additionally, these methods lack the ability to provide explanatory information regarding what structural features contribute to the observed chemical activity.

Current work on kernel functions is limited by similar issues stemming from the high complexity of graph objects. The use of complex structures such as general subgraphs in comparing objects gives kernel functions more expressive power, but the computational cost of such detailed comparison renders these methods intractable for the large databases that are now available for analysis. Using simpler graph features such as sets and sequences can significantly reduce the computational burden, but at the price of model depth and power. The tradeoff is clear, and while the groundwork for many classes of graph kernel functions has been laid, there is still no satisfactory middle ground that combines high discriminative ability with fast computation speed.

The work presented in this thesis explores some graph kernel functions that improve on existing methods with respect to both classification accuracy and kernel computation time. The following key insights are explored. First, problem relevant structure features can be used to annotate graph vertices in an alignment-based kernel function, raising model accuracy and adding explanatory capability [45]. Second, extensions for matching approximate structure features [42], as well as a faster, simpler kernel function [43] lead to gains in accuracy as well as faster computation time. Finally, wavelet functions can be applied to graphs in order to summarize feature information in local graph topology, greatly reducing the kernel computation time [44]. These methods are validated using a series a chemical structure-activity data sets, such as prediction of protein-chemical binding affinity, toxicity, and intestinal absorption.

This remainder of the text is organized as follows. Chapter 2 introduces a variety of background material regarding graphs, chemicals, graph pattern mining, kernel functions, and wavelets. In Chapter 3 the previous related work in graph classification and kernel functions is reviewed. The next four chapters, 4 though 7, describe the algorithmic details of this work's contributions to graph classification and kernel functions, and analyze their merits in experimental studies. Some of the material in these chapters is duplicated, but was included, despite redundancy, to facilitate ease of reading. Finally, Chapter 8 discusses the overall conclusions and insights gained from this work, as well as possible future work.

Chapter 2

Background

Before proceeding to algorithmic details, this chapter presents some general background material from a variety of directions. The work of this thesis draws from techniques in data mining as well as machine learning and chemical property prediction. This chapter will address the following topics: chemical structures as graphs, graph classification, kernel functions, graph mining, and wavelet analysis for graphs.

2.1 Chemical Structure

Chemical compounds are organic molecules that are easily modeled by a graph representation. In this approach, *nodes* in a graph model *atoms* in a chemical structure and *edges* in the graph to model chemical *bonds* in the chemical structure. In this representation, nodes are labeled with the atom element type, and edges are labeled with the bond type (single, double, and aromatic bond). The edges in the graph are undirected, since there is no directionality associated with chemical bonds. Figure 2.1 shows an example chemical structure, where unlabeled

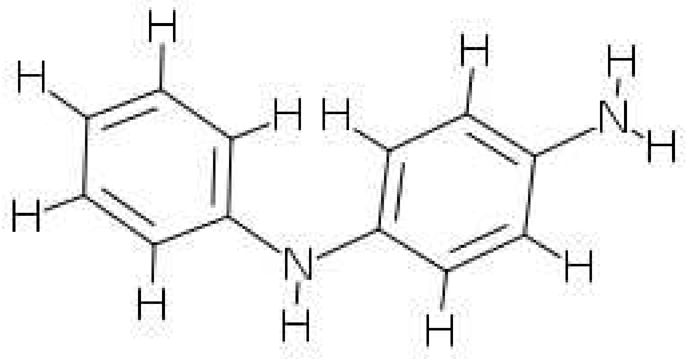


Figure 2.1. An example chemical structure.

vertices are assumed to be carbon (C).

Figure 2.1 shows three sample chemical structures on the left, and their graph representation on the right.

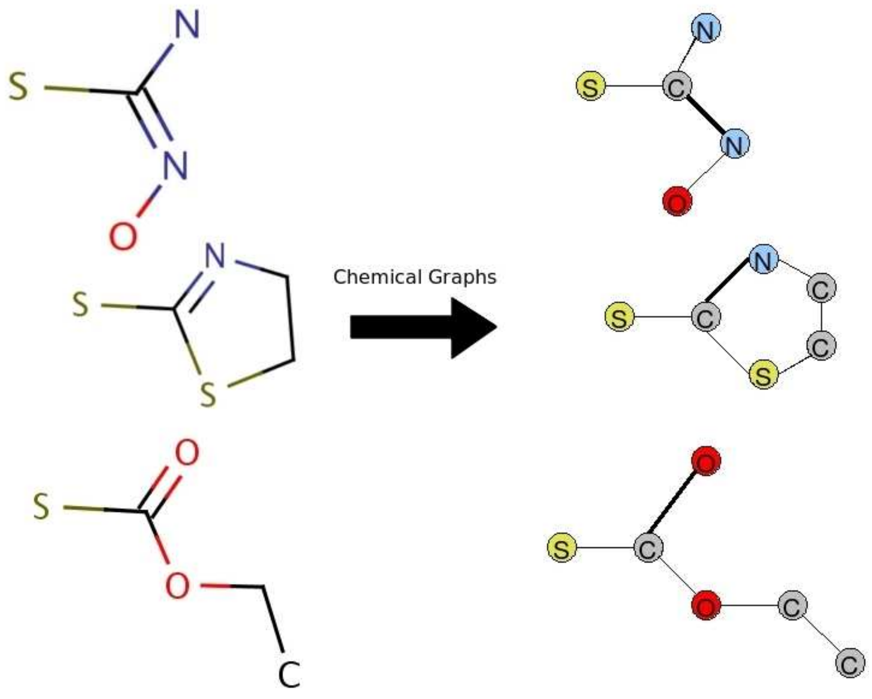


Figure 2.2. Graph representations of chemicals.

2.2 Graph Classification

Many classifiers exist for classification of objects as feature vectors. The feature vector embeds objects as points in a space where the data is modeled. Recently an important linear classifier has gained a great deal of attention, the Support Vector Machine (SVM). It is not only fast to train with great model generalization power, but it is also a kernel classifier giving it additional advantages over establish vector space classifiers. These issues will be addressed in the following section on kernel functions.

SVM builds a classification model by finding a linear hyperplane that best separates the classes of data objects. The *optimal separating hyperplane* (OSH) is chosen by maximizing the margin between the the hyperplane and the nearest data points (termed support vectors).

When data are not linearly separable, called the soft-margin case, the SVM finds a hyperplane that optimizes an additional constraint. Often this constraint is a penalty for misclassified samples expressed in various ways.

The problem of finding an OSH is formulated as a convex optimization problem and hence can leverage very powerful algorithms for finding exactly finding the OSH. Once a OSH has been found, classification of additional objects is easily determined by finding which side of the hyperplane the object resides on. The efficiency of these operations makes SVM an extremely fast classifier. Since the SVM model ideally depends only on a small number of support vectors it generalizes well and is compact to store.

Crucially, the SVM problem can be formulated such that it represents objects using only the dot products between their vectors. This modification allows the dot products to be replaced with a kernel function between objects, the use of which is discussed further in the follow section.

2.3 Kernel Functions

A kernel function K is a mapping between a pair of graphs into a real number, $K : GxG \rightarrow \mathbb{R}$. This function defines an inner product between two graphs and must satisfy the following properties.

- Positive semi-definite. $\sum_i \sum_j K(g_i, g_j)c_i c_j \geq 0, \forall g \in G, \forall c \in \mathbb{R}$.
- Symmetric. $K(g_i, g_j) = K(g_j, g_i), \forall g \in G$.

Such a function embeds graphs or any other objects into a Hilbert space, and is termed a Mercer kernel from Mercer's theorem.

Kernel functions can enhance classification in two ways: first, by mapping vector objects into higher dimensional spaces; second, by embedding non-vector objects in an implicitly defined space. The advantages of mapping objects into a higher dimensional space, the so called *kernel trick* are apparent in a variety of cases where objects are not separable by a linear decision boundary.

This implicit embedding is not only only useful for non-linear mappings, but also serves to decouple the object representation from the spatial embedding. A kernel function need only be defined between data objects in order to apply SVM classification. Therefore SVM can be used for classification of graph objects by defining a kernel function between graphs, without explicitly defining any set of graph features.

2.4 Graph Database Mining

This section discusses a few important definitions for graph database mining: labeled graphs, subgraph isomorphic relation, and graph classification.

Definition 2.4.1 *A* ***labeled graph*** *G is a quadruple $G = (V, E, \Sigma, \lambda)$ where V is a set of vertices or nodes and $E \subseteq V \times V$ is a set of undirected edges. Σ is a set of (disjoint) vertex and edge labels, and λ: $V \cup E \rightarrow \Sigma$ is a function that assigns labels to vertices and edges. Assume that a total ordering is defined on the labels in Σ.*

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

Definition 2.4.2 *A graph $G' = (V', E', \Sigma', \lambda')$ is* ***subgraph isomorphic*** *to $G = (V, E, \Sigma, \lambda)$, denoted by $G' \subseteq G$, if there exists a 1-1 mapping $f : V' \rightarrow V$ such that*

- $\forall v \in V', \lambda'(v) = \lambda(f(v))$
- $\forall (u, v) \in E', (f(u), f(v)) \in E$, *and*
- $\forall (u, v) \in E', \lambda'(u, v) = \lambda(f(u), f(v))$

.

The function f is a *subgraph isomorphism* from graph G' to graph G. It is said G' *occurs* in G if $G' \subseteq G$. Given a subgraph isomorphism f, the image of the domain V' ($f(V')$) is an *embedding* of G' in G.

Example 2.4.1 *Figure 2.3 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*

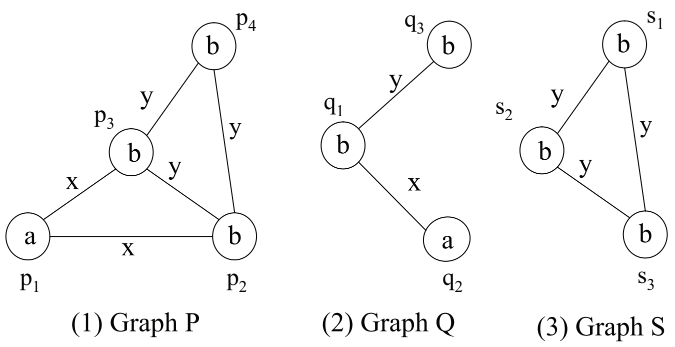


Figure 2.3. A Database of three labeled graphs.

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 of these graphs $D = \{(G_i, T_i,)\}_{i=1}^{n}$, the **graph classification problem** is to estimate a function $F : G^* \rightarrow T$ that accurately map graphs to their target value.

By *classification* all target values are assumed to be discrete values, otherwise it is a *regression* problem. Below, several algorithms are reviewed for graph classification that work within a common framework called a kernel function. 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.

2.5 Wavelet Analysis for Graphs

Wavelet functions are commonly used as a means for decomposing and representing a function or signal as its constituent parts, across various resolutions or scales. Wavelets are usually applied to numerically valued data such as communication signals or mathematical functions, as well as to some regularly structured numeric data such as matrices and images.

Graphs, however, are arbitrarily structured and may represent many relationships and topologies between data elements. Recent work has established the successful application of wavelet functions to graphs for multi-resolution analysis [5]. The use of wavelets in this capacity is different than the use of wavelets for signal and image compression such as in [32]. The complex graph topology must be projected into a Euclidean space, and wavelets are used to summarize the information in the local topology around graph nodes.

Given a vertex v in graph G, define the h-hop neighbors of v as the set of other nodes in G whose shortest path to v is h nodes. The sets of h-hop neighbors then lead to the notion of hop distance which suitably projects the nodes of a graph into Euclidean space.

Wavelets are then used to summarize feature information in the local topology around vertices in a graph. Since regions near the origin in a wavelet function a strongly positive, while the regions farther away are strongly negative, with distant regions neutral, using a wavelet function to compute a weighted sum over vertex features arranged according to hop distance corresponds to a comparison of vertex features in the local neighborhood to those in the distant neighborhood.

Chapter 3

Related Works

Given that graphs are such powerful and interesting structures, their classification has been extensively studied. This chapter will review the related work covering pattern mining, kernel functions, and wavelets for graph analysis.

This section surveys 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. The possible features include both structural and chemical. Structural features are graph fragments of different types. Examples are paths, cycles, trees, and general subgraphs [60]. Chemical descriptors, as they are called in QSAR work, are properties describing a molecule overall such as weight, charge.

Once a set of features is determined, a graph is described by a feature vector, and any existing classification methods such as CBA [3] and decision tree [38] that work in an n-dimensional Euclidian space, may be applied for graph classification.

The second approach to classification 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* [51].

The advantage of a kernel method is that it has a lower chance of over fitting, which is a serious concern in high dimensional space with low sample size.

The following sections summarize recent work related to pattern mining and structural features, as well as vector-based classification , kernel functions for classification, and wavelets for graphs.

3.1 Pattern Mining

Algorithms that search for frequent patterns (e.g. trees, paths, cyclic graphs) in graphs can be roughly divided into three groups.

The first group uses a level-wise search strategy, including AGM [20] and FSG [28]. The second category takes a depth-first search strategy, including gSpan [58] and FFSM [21]. Different from level-wise search algorithms AGM and FSG, the depth-first search strategy utilizes a back-track algorithm to mine frequent subgraphs. The advantage of a depth-first search is a better memory utilization since depth-first search keeps one frequent subgraph in memory and enumerates its supergraphs, in contrast to keeping all k-edge frequent subgraph in memory.

The third category of frequent subgraph mining algorithms does not work directly on a graph space to identify frequent patterns. Instead, 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. This strategy is called *progressive mining.* Algorithms in this category includes SPIN [18] and GASTON [34].

3.1.1 Frequent Subgraphs

Frequent subgraph mining is a technique used to enumerate graph substructures that occur in a graph database with at least some specified frequency. This minimum frequency threshold is termed the *support threshold* by the data mining community. After limiting returned subgraphs by frequency, types found can be further constrained by setting upper and lower limits on the number of vertices they can contain. In much the work of this thesis, the FFSM algorithm [17] is used for fast computation of frequent subgraphs. Figure 3.1.1, adopted from [17], shows an example of this frequent subgraph enumeration. Some work has been done by Deshpande et al. [7] toward the use of these frequent substructures in the classification of chemical compounds with promising results.

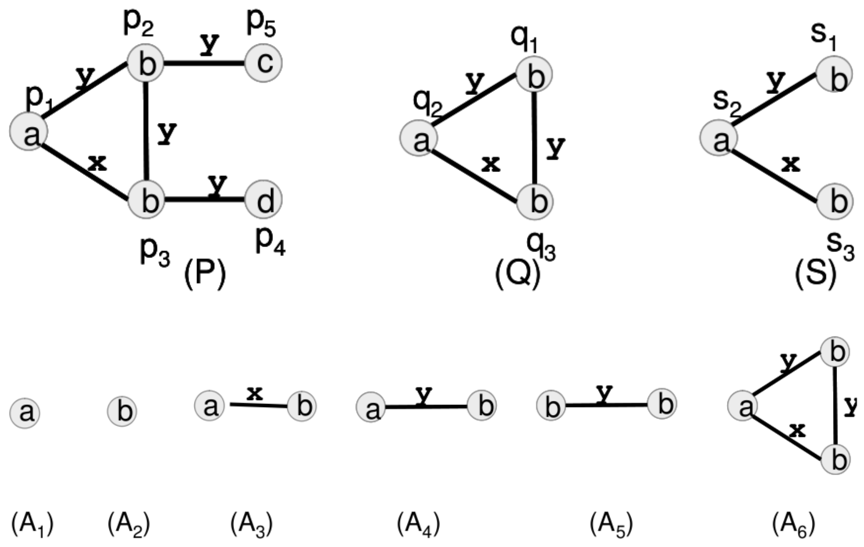


Figure 3.1. Example graphs and frequent subgraphs (support = 2/3).

3.1.2 Chemical Properties and Target Prediction

A *target property* of the chemical compound is a measurable quantity of the compound. There are two categories of target properties: continuous (e.g., binding affinities to a protein) and discrete target properties (e.g. active compounds vs. inactive compounds).

The relationship between a chemical compound and its target property is typically investigated through a quantitative structure-property relationship (*QSPR*) [1]. Abstractly, any QSPR method may be generally defined as a function that maps a chemical space to a property space in the form of

$$P = \hat{k}(D) \tag{3.1}$$

where D is a chemical structure, P is a property, and the function $\hat{k}$ is an estimated mapping from a chemical space to a property space.

Different QSPR methodologies can be understood in terms of the types of target property values (continuous or discrete), types of features, and algorithms that map descriptors to target properties.

3.2 Vector-based Classification

Several classification algorithms based on explicitly collected features exist for graph classification in a variety of applications. What follows is a brief survey of popular methods for some pertinent applications

Recent methods applied to QSAR and chemical activity prediction include Decision Trees, Classification based on association [3], and Random Forest among

[1]Such study also known as a quantitative structure-activity relationship (QSAR) but *property* refers to a broader range of applications than activity.

many others. Decision trees use a collection of simple learners organized in a hierarchical tree structure to classify a object. Non-leaf nodes makes decisions about an object based on one of it's properties and send it to one of the children. Leaf nodes of the tree correspond to classification categories. Random forest uses a collection of randomly generated decision trees and typically classify an object according to the mode of the classes returned by all trees.

Classification based on association (CBA) is somewhat different than these other methods. CBA seeks to find a set of association rules of the form $A \rightarrow c_i$, where A is some set of properties and c_i is a class label. XRules [60] is similar to CBA and 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 $p(c_i|G)$ estimated from the training data. XRules carefully selects a subset of rules with high confidence values and uses those rules for classification.

Graph boosting [27] 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.

3.3 Kernel Functions for Graph Classification

The term kernel function refers to an operation for computing the inner product between two vectors in a feature space, thus avoiding the explicit computation of coordinates in that feature space. Graph kernel functions are simply kernel functions that have been defined to compute the similarity between two graph structures. In recent years a variety of graph kernel functions have been developed, with promising results as described by Ralaivola et al [29].

Graph kernel functions can 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 [12], random walk based kernels [24], and kernels based on shortest paths between pair of nodes [26]. 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 [40], cyclic kernels [48], spectrum kernel [7]. This section reviews the relevant work on these kernel functions.

Product graph kernels use a feature space of all possible node label sequences for walks in graphs. Since the number of possible walks are infinite, there is no way to enumerate all the features in kernel computation [12]. Instead, a *product graph* is computed in order to make the kernel function computation feasible.

Rather than computing the shared paths exactly, which has prohibitive computational cost for large graphs, the work of Kashima et al. [24] is based on the use of shared random label sequences in the computation of graph kernels. Their marginalized graph kernel uses a Markov model to randomly generate walks of a labeled graph. The random walks are created using 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 molecules.

Spectrum kernels aim to simplify the aforementioned kernels by working in a finite dimensional feature space based on a set of subgraphs (or as special cases, trees, cycles, and paths). The kernel function is computed as the inner product between two feature vectors, such as counts of subgraph occurrences as in [7]. Transformations of the inner product, such as min-max kernel [52] and Tanimoto kernel [29], are also widely used. The subtree kernel [33] is a variation on the spectrum kernel that uses subtrees instead of paths.

The optimal assignment kernel, described by Frölich et al [10], differs significantly from the marginalized graph kernel. This kernel function first computes the similarity between all vertices in one graph and all vertices 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.

3.4 Wavelets Functions for Graphs

Crovella et al. [5] have developed a multi-scale method for network traffic data analysis. For this application, they are attempting to determine the scale at which certain traffic phenomena occur. They represent traffic networks as graphs labeled with some measurement such as bytes carried per unit time. In their method, they use the *hop* distance between vertices in a graph, defined as the length of the shortest path between them, and apply a weighted average function to compute the difference between the average of measurements close to a vertex and measurements that are far, up to a certain distance. This process produces a new measurement for a specific vertex that captures and condenses information about the vertex neighborhood. Figure 3 shows a diagram of wavelet function weights overlayed on a chemical structure.

Maggioni et al. [32] demonstrate a general-purpose biorthogonal wavelet for graph analysis. In their method, they use the dyadic powers of an diffusion operator to induce a multiresolution analysis. While their method applies to a large class of spaces, such as manifolds and graphs, the applicability of their method to attributed chemical structures is not clear. The major technical difficulty is how to incorporate node labels in a multiresolution analysis.

Chapter 4

Alignment Kernels with Pattern-based Features

Traditional approaches to graph similarity rely on the comparison of compounds using a variety of molecular attributes known *a priori* to be involved in the activity of interest. Such methods are problem-specific, however, and provide little assistance when the relevant descriptors are not known in advance. Additionally, these methods lack the ability to provide explanatory information regarding what structural features contribute to the observed chemical activity. The method proposed here, referred to as OAPD for Optimal-Assignment with Pattern-based Descriptors, alleviates both of these issues through the mining and analysis of structural patterns present in the data in order to identify highly discriminating patterns, which then augment a graph kernel function that computes molecular similarity.

4.1 Structure-based Pattern Mining For Chemical Compound Classification

The following sections outline the algorithm that drives the experimental method. In short, it measures the similarity of graph structures whose vertices and edges have been labeled with various descriptors. These descriptors represent physical and chemical information such as atom and bond types. They are also used to represent the membership of atoms in specific structure patterns that have been mined from the data. To compute the similarity of two graphs, the vertices of one graph are aligned with the vertices of the second graph, such that the total overall similarity is maximized with respect to all possible alignments. Vertex similarity is measured by comparing vertex descriptors, and is computed recursively so that when comparing two vertices, it also compares the neighbors of those vertices, and their neighbors, etc.

4.1.1 Structure Pattern Mining

The frequent subgraph mining problem can be phrased as such: given a set of labeled graphs, the support of an arbitrary subgraph is the fraction of all graphs in the set that contain that subgraph. A subgraph is frequent if its support meets a certain minimum threshold. The goal is to enumerate all the frequent, connected subgraphs in a graph database. The extraction of important subgraph patterns can be controlled by selecting the proper frequency threshold, as well as other parameters such as size and density of subgraph patterns.

4.1.2 Optimal Assignment Kernel

The optimal assignment kernel function computes the similarity between two graph structures. This similarity computation is accomplished by first representing the two sets graph vertices as a bipartite graph, and then finding the set of weighted edges assigning every vertex in one graph to a vertex in the other. The edge weights are calculated via a recursive vertex similarity function. This section presents the equations describing this algorithm in detail, as discussed by Frölich et al [9]. The top-level equation describing the similarity of two molecular graphs is:

$$k_A(M_1, M_2) := max_\pi \sum_{h=1}^{m} k_{nei}(v_{\pi(h)}, v_h) \tag{4.1}$$

Where π denotes a permutation of a subset of graph vertices, and m is the number of vertices in the smaller graph. This is needed to assign all vertices of the smaller graph to vertices in the large graph. The k_{nei} function, which calculates the similarity between two vertices using their local neighbors, is given as follows:

$$k_{nei}(v_1, v_2) := k_v(v_1, v_2) + R_0(v_1, v_2) + S_{nei}(v_1, v_2) \tag{4.2}$$

$$S_{nei}(v_1, v_2) := \sum_{l=1}^{L} \gamma(l) R_l(v_1, v_2) \tag{4.3}$$

The functions k_v and k_e compute the similarity between vertices (atoms) and edges (bonds), respectively. These functions could take a variety of forms, but in the OA kernel they are RBF functions between vectors of vertex/edge labels.

The $\gamma(l)$ term is a decay parameter that weights the similarity of neighbors according to their distance from the original vertex. The l parameter controls the topological distance within which to consider neighbors of vertices. The R_l equation, which recursively computes the similarity between two specific vertices is given by the following equation:

$$R_l(v_1, v_2) = \frac{1}{|v_1||v_2|} \sum_{i,j} R_{l-1}(n_i(v_1), n_j(v_2)) \tag{4.4}$$

Where $|v|$ is the number of neighbors of vertex v, and $n_k(v)$ is the set of neighbors of v. The base case for this equation is R_0, defined by:

$$R_0(v_1, v_2) := \frac{1}{|v_1|} max_\pi \sum_{i=1}^{|v_2|} (k_v(a, b)|k_e(x, y)) \tag{4.5}$$

$$a = n_{\pi(i)}(v_1),\ b = n_i(v_2) \tag{4.6}$$

$$x = v_1 \rightarrow n_{\pi(i)}(v_1),\ y = v_2 \rightarrow n_i(v_2) \tag{4.7}$$

The notation $v \rightarrow n_i(v)$ refers to the edge connecting vertex v with the ith neighboring vertex. The functions k_v and k_e are used to compare vertex and edge descriptors, by counting the total number of descriptor matches.

4.1.3 Reduced Graph Representation

One way in which to utilize the structure patterns that are mined from the graph data is to collapse the specific subgraphs into single vertices in the original graph. This technique is explored by Frölich et al. [10] with moderate results, although they use predefined structure patterns, so called pharmacophores, identified *a priori* with the help of expert knowledge. The method proposed here ushers these predefined patterns in favor of the structure patterns generated via frequent subgraph mining.

The use of a reduced graph representation does have some advantages. First, by collapsing substructures, an entire set of vertices can be compared at once, reducing the graph complexity and marginally decreasing computation time. Second, by changing the substructure size the resolution at which graph structures are compared can be adjusted. The disadvantage of a reduced graph representation is that substructures can only be compared directly to other substructures, and cannot align partial structure matches. As utilized in Frölich et al., this is not as much of a burden since they have defined the best patterns *a priori* using expert knowledge. In the case of the method presented here, however, this is a significant downside, as there is no *a priori* knowledge to guide pattern generation and we wish to retain as much structural information as possible.

4.1.4 Pattern-based Descriptors

The loss of partial substructure alignment following the use of a reduced graph representation motivated us to find another way of integrating this pattern-based information. Instead of collapsing graph substructures, vertices are simply annotated with additional descriptor labels indicating the vertex's membership in the structure patterns that were previously mined. These pattern-based descriptors are calculated for each vertex and are used by the optimal assignment kernel in the same way that other vertex descriptors are handled. In this way substructure information can be captured in the graph vertices without needing to alter the original graph structure.

4.2 Experimental Study

Classification experiments were conducted on five different biological activity data sets, and measured support vector machine (SVM) classifier prediction accuracy for several different feature generation methods. The data sets and classification methods are described in more detail in the following subsections, along with the associated results. Figure 4.1 gives a graphical overview of the process.

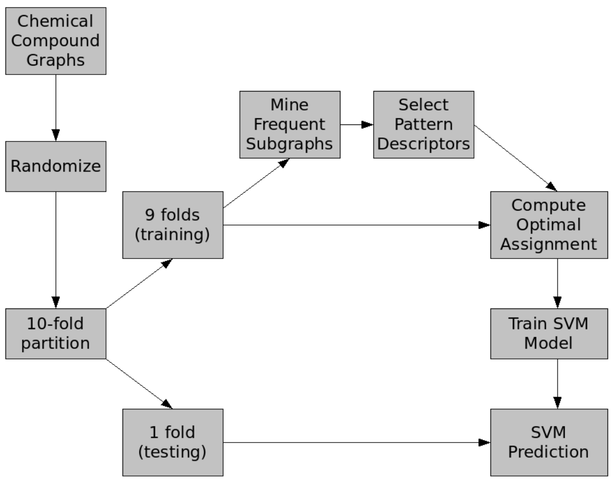


Figure 4.1. Experimental workflow for a cross-validation trial with frequent subgraph mining.

All of the experiments were performed on a desktop computer with a 3Ghz Pentium 4 processor and 1 GB of RAM. Generating a set of frequent subgraphs is very quick, generally a few seconds. Optimal assignment requires significantly more computation time, but not intractable, at less than half an hour for the largest data set.

4.2.1 Data Sets

Five data sets used in various problem areas were selected to evaluate classifier performance. The Predictive Toxicology Challenge data set, discussed by Helma et al [14], contains a set of chemical compounds classified according to their toxicity in male rats (PTC-MR), female rats (PTC-FR), male mice (PTC-MM), and female mice (PTC-FM). The Human Intestinal Absorption (HIA) data set (Wessel et al. [53]) contains chemical compounds classified by intestinal absorption activity. Also included were two different virtual screening data sets (VS-1,VS-2) used to predict various binding inhibitors from Fontaine et al. [8] and Jorissen et al [23]. The final data set (MD) is from Patterson et al [35], and was used to validate certain molecule descriptors. Various statistics for these data sets can be found in Table 4.1.

Table 4.1. Data set statistics for OAPD experiments.

Dataset	Number of Compounds	Number of Positives	Number of Negatives	Average Compound Size
HIA	86	47	39	22.45
MD	310	148	162	10.38
VS-1	435	279	156	59.81
VS-2	1071	125	946	39.93
PTC-MR	344	152	192	25.56
PTC-MM	336	129	207	25.05
PTC-FR	351	121	230	26.08
PTC-FM	349	143	206	25.25

4.2.2 Methods

The performance of the SVM classifier was evaluated by training with several different feature sets. The first set of features (FSM) consists only of frequent subgraphs. Those subgraphs are mined using the FFSM software [17] with minimum subgraph frequency of 50%. Each chemical compound is represented by a binary vector with length equal to the number of mined subgraphs. Each subgraph is mapped to a specific vector index, and if a chemical compound contains a subgraph then the bit at the corresponding index is set to one, otherwise it is set to zero.

The second feature set (OA) consists of the similarity values computed by the optimal assignment kernel, as proposed by Frölich et al. Each compound is represented as a real-valued vector containing the computed similarity between it and all other molecules in the data set.

The third feature set (OARG) is computed using the optimal assignment kernel as well, except that the frequent subgraph patterns are embedded as a reduced graph representation before computing the optimal assignment. The reduced graph representation is described by Frölich et al. as well, but they use *a priori* patterns instead of frequently mined ones.

Finally, the fourth feature set (OAPD) also consists of the subgraph patterns combined with the optimal assignment kernel, however in this case a reduced graph is not derived, and instead annotate vertices in a graph with additional descriptors indicating its membership in specific subgraph patterns.

In the experiments, support vector machine (SVM) classifier was used in order to generate activity predictions. The use of SVM has recently become quite popular for a variety of biological machine learning applications because of its efficiency and ability to operate on high-dimensional data sets. The SMO SVM classifier was used, implemented by Platt [37] and included in the Weka data-mining software package by Witten et al [56]. The SVM parameters were fixed, with a linear kernel and $C = 1$. Classifier performance was averaged over a ten-fold cross-validation set.

Some feature selection was performed in order to identify the most discriminating frequent patterns. Using a simple statistical formula, known as the Pearson correlation coefficient (PCC), the correlation between a set of feature samples (in this case, the occurrences of a particular subgraph in each of the data samples) and the corresponding class labels was measured. Frequent patterns are ranked according to correlation strength, and the top patterns are selected.

4.2.3 Results

Table 4.2 contains results reporting the average and standard deviation of the prediction accuracy over the 10 cross-validation trials. From the table, the following observations can be made.

Table 4.2. Average and standard deviation of 10-fold cross-validation accuracy for OAPD experiments.

Dataset	Method			
	FSM	OA	OARG	OAPD
HIA	57.36 ±19.11	63.33 ±20.82	62.92 ±22.56	65.28 ±15.44
MD	68.39 ±7.26	70.00 ±6.28	69.35 ±6.5	70.32 ±5.65
VS-1	60.00 ±5.23	64.14 ±3.07	62.07 ±4.06	63.91 ±4.37
VS-2	90.29 ±2.3	94.96 ±1.88	93.18 ±2.68	94.77 ±2.17
PTC-FM	54.16 ±5.82	61.35 ±9.53	59.03 ±6.46	59.29 ±8.86
PTC-FR	63.28 ±5.32	60.10 ±9.21	64.68 ±3.96	64.39 ±3.6
PTC-MM	60.45 ±3.87	62.16 ±6.43	62.75 ±7.69	63.05 ±5.24
PTC-MR	58.42 ±4.43	56.41 ±6	54.07 ±7.52	60.76 ±7.32

First, notice that OAPD (and OARG) outperforms FSM methods in all of the tried data sets except one (FSM is better than OARG on the PTC-MR data set). This results indicate that use of frequent subgraphs alone without using the optimal alignment kernel, does not produce a good classifier. Although the conclusion is generally true, interestingly, for the PTC-MR data set, the FSM method outperforms both the OA and OARG methods, while the OAPD method outperforms FSM. This seems to suggest that important information is encoded in the frequent subgraphs, and is being lost in the OARG, but is still preserved in the OAPD method.

Second, notice that OAPD (or OARG) method outperforms the original OA method in 5 of the tried 8 data sets: HIA, MD, PTC-FR, PTC-MM, PTC-MR. OAPD has a very close performance to that of OA in the rest of the three data sets. The results indicate that the OAPD method provides good performance for diverse data sets which involve tasks such as predicting chemical's toxicology, predicting human intestinal absorption of chemicals, and virtual screening of drugs.

Table 4.3. SMARTS string of highly ranked chemical patterns from OAPD method.

HIA	MD	VS-1	VS-2
[NH3+]C(C)C	C(=CC)(C)S	C(=CC=C)C=C	C(=CCC)C
C(=C)(C)C	C(=CC=CC)(C)S	C(=CC)CNC	C=CCC
C(=CC)(C)C	C(=C)(C=CC=C)S	C(=C)CNC	[NH2+](CC=C)CC
C(=CC)(C=C)C	C(=CCC)C=C	CC(=CC)N	[NH2+](CCC)CC
C(=CC=C)(C=C)C	C(=CS)C=C	CNCC=CC	[NH3+]CC(=CC)C

PTC-MR	PTC-MM	PTC-FR	PTC-FM
[NH2+]C(=C)C=C	[NH3+]CC	[NH2+]C(=CC)C=C	OCC=C
[NH2+]C=CC	c1ccccc1	[NH2+]C(=C)C=C	C(=CC)C(=C)C
[NH3+]CC	C(=CC)C(=C)C	[NH3+]CC	CCC=CC
CC=C	C(=CC=C)C	CC=C	C(=C)(C)C
C(CC)C	C(=C)C(=C)C	C(CC)C	c1ccccc1

In addition to outperforming the previous methods, this method also reports the specific subgraph patterns that were mined from the training data and used to augment the optimal assignment kernel function. By identifying highly discriminative patterns, this method can offer additional insight into the structural features that contribute to a compound's chemical function. Table 4.3 contains the five highest ranked (using Pearson correlation coefficient) subgraph patterns for each data set, expressed as SMARTS strings that encode the specific pattern. Many of the patterns in all sets denote various carbon chains (C(CC)C, C=CC, etc.), however there seem to be some unique patterns as well. The MD data set contains carbon chain patterns with some sulfur atoms mixed in, while the VS-1 data set has carbon chains with nitrogen mixed in. The [NH2+] and [NH3+] patterns appear to be important in the VS-2 data set, as well as some of the PTC data sets.

4.3 Conclusions

Graph structures are a powerful and expressive representation for chemical compounds. This work presents a new method, termed OAPD, for computing the similarity of chemical compounds, based on the use of an optimal assignment graph kernel function augmented with pattern-based descriptors that have been mined from a set of molecular graphs. Experimental studies demonstrate that the OAPD method integrates the structural alignment capabilities of the existing optimal alignment kernel method with the substructure discovery capabilities of the frequent subgraph mining method and delivers better performance in most of the tried benchmarks. In the future, it may be possible to involve domain experts to evaluate the performance of this algorithm, including the prediction accuracy and the capability of identifying structure important features, in diverse chemical structure data sets.

Chapter 5

Alignment Kernels with Approximate Pattern Features

The work presented in this chapter aims to leverage existing frequent pattern mining algorithms and explore the application of kernel classifiers in building highly accurate graph classification algorithms. Towards that end, a novel technique is demonstrated called graph pattern diffusion kernel (GPD). In this method, all frequent patterns are first identified from a graph database. Then subgraphs are mapped to graphs in the graph database and nodes of graphs are projected to a high dimensional space with a specially designed function. Finally a novel graph alignment algorithm is used to compute the inner product of two graphs. This algorithm is tested using a number of chemical structure data sets. The experimental results demonstrate that this method is significantly better than competing methods such as those based on paths, cycles, and other subgraphs.

5.1 GPD: A Graph Pattern Diffusion Kernel for Accurate Graph Classification

Here the design of the pattern diffusion kernel is presented. The section begins by first presenting a general framework. It is proved, through a reduction to the subgraph isomorphism problem, that the computational cost of the general framework can be prohibitive for large graphs. The pattern based graph alignment kernel is then presented. Finally a technique is shown called "pattern diffusion" that can significantly improve graph classification accuracy in practice.

5.1.1 Graph Similarity Measurement with Alignment

An *alignment* of two graphs G and G' (assuming $|V[G]| \leq |V[G']|$) is a 1-1 mapping $\pi : V[G] \rightarrow V[G']$. Given an alignment π, define the similarity between two graphs, as measured by a kernel function k_A, below:

$$k_A(G, G') = \max_{\pi} \sum_{v} k_n(v, \pi(v)) + \sum_{u,v} k_e((u, v), (\pi(u), \pi(v))) \tag{5.1}$$

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 5.1 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.

5.1.2 NP-hardness of Graph Alignment Kernel Function

It is no surprise that computing the graph alignment kernel is an NP-hard problem. It is proved this with a reduction from the graph alignment kernel to the subgraph isomorphism problem. In the following paragraphs, assuming there exists an efficient solver of the graph alignment kernel problem, it is shown 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 mentioned before, it is proved that the graph alignment kernel problem is therefore an NP-hard problem as well. Note: this subsection is a stand-alone component of this work, and readers who choose to skip this section should encounter no difficulty in reading the rest of the text.

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), use a node kernel function that returns a constant 0. Define an edge kernel function $k_e : V[G] \times V[G] \times V[G'] \times V[G'] \to \mathbb{R}$ as

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

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

$$k_A(G,G') = \max_{\pi} \sum_{u,v} k_e((u,v),(\pi(u),\pi(v))) \tag{5.2}$$

The NP-hardness of the graph alignment kernel is established with the following theorem.

Theorem 5.1.1 *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 5.1.1 *If: notice from the definition of k_e that the maximal value of $K_a(G, G')$ is $|E[G]|$. Given $K_a(G, G') = |E[G]|$, it is claimed that there exists an alignment function $\pi : V[G] \rightarrow V[G']$ such that for all $(u, v) \in E[G]$, $(\pi(u), \pi(v)) \in E[G']$. 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', there is an alignment function $\pi : V[G] \rightarrow V[G']$ such that for all $(u, v) \in E[G]$, $(\pi(u), \pi(v)) \in E[G']$. According to Equation 5.2, $K_a(G, G') = |E[G]|$.

Theorem 5.1.1 shows that the graph alignment kernel problem is no easier than the subgraph isomorphism problem and hence is at least NP-hard in complexity.

5.1.3 Graph Node Alignment Kernel

To derive an efficient algorithm scalable to large graphs, the idea is that a function f is used 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 such a function f is obtained, the graph kernel function may be simplified with the following formula:

$$k_M(G, G') = \max_{\pi} \sum_{v \in V[G]} k_n(f(v), f(\pi(v))) \tag{5.3}$$

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, a weighted complete bipartite graph is constructed 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 $k_n(f(v), f(u))$. Figure 5.1, shows a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$. Highlighted edges $(v1, u2)$, $(v2, u1)$, $(v3, u3)$ have larger weights than the rest of the edges (dashed).

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. The Hungarian algorithm is used with complexity $O(|V[G]|^3)$. For details of the Hungarian algorithm see [1].

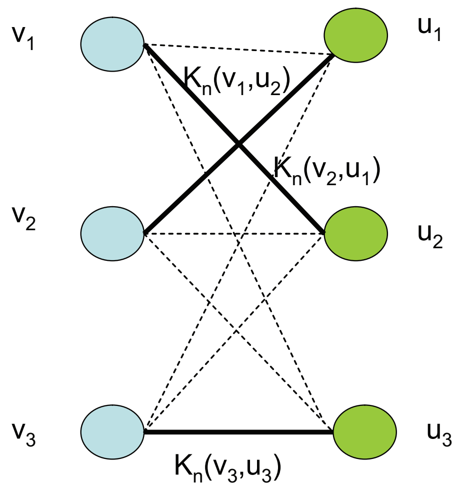


Figure 5.1. A maximum weighted bipartite graph for graph alignment.

Applying the Hungarian algorithm to graph alignment was first explored by [10] 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, a new framework is developed here 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, experiments show that this technique generates similar and sometimes better classification accuracies compared to the method reported in [10].

Unfortunately, using the Hungarian algorithm for assignment, as used by [10] is not a true Mercer kernel. Since the kernel function proposed here uses this algorithm as well, it is also not a Mercer kernel. Like in [10], however, practically this kernel still performs competitively.

5.1.4 Pattern Diffusion

This section introduces a novel technique "pattern diffusion" to project nodes in a graph to a high dimensional space that captures both node labeling information and local topology information. This design has the following advantages as a kernel function:

- The 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.
- It is proved 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, the pattern diffusion kernel is outlined in three steps.

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

In the second step given a set of nodes S as seed, 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, define f_{t+1} $(t \geq 0)$ with f_t in the following way:

$$f_{t+1}(v) = f_t(v) \times (1 - \frac{\lambda}{d(v)}) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)} \tag{5.4}$$

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.

The formula 5.4 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. Call it "diffusion" because the process simulates the way a value is spreading in a network. The 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, one parameter called diffusion time is used, denoted by τ, to control the diffusion process. Specifically the diffusion process is limited 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, define the mapping function f_S as the limit function of f_t as t approaches to infinity, or

$$f_S = \lim_{t \to \infty} f_t \tag{5.5}$$

5.1.5 Pattern Diffusion Kernel and Graph Classification

This section summarizes the discussion of kernel functions and shows how they are utilized to construct an efficient graph classification algorithm at both the training and testing phases.

5.1.5.1 Training Phase

In the training phase, 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, there are two groups of graphs: positive or negative. For multi-class classification, there are 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. Identify frequent subgraphs from each graph group and union the subgraph sets together as the seed set $\mathcal{S}$.
- For each seed $S \in \mathcal{S}$ and for each graph G in the training data set, 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 $m = |\mathcal{S}|$.
- For two graphs G, G', construct the complete weighted bipartite graph as described in section 5.1.3 and compute the kernel $K_a(G, G')$ using Equation 5.3.

- Train a predictive model using a kernel classifier.

5.1.5.2 Testing Phase

In the testing phase, the kernel function is computed for graphs in the testing and training data sets. The trained model is used to make predictions about graph in the testing set.

- For each seed $S \in \mathcal{S}$ and for each graph G in the testing data set, f_S is used to label nodes in G and create feature vectors as done in the training phase.
- Equation 5.3 computes 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

The empirical study of different kernel functions, including the pattern diffusion kernel, is presented below.

5.2 Experimental Study

Classification experiments were conducted using ten different biological activity data sets, and compared cross-validation accuracies for different kernel functions. The following subsections describe the data sets and the classification methods in more detail along with the associated results.

All of experiments were performed on a desktop computer with a 3Ghz Pertium 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, the LibSVM software [4] was used as the kernel classifier. The nu-SVC type classifier was used with nu = 0.5, the LibSVM default. To perform a fair comparison, model selection was not performed, and the SVM parameters were not tuned to favor any particular method, and default parameters were used in all cases. The classifiers CBA and Xrule were downloaded as instructed in the related papers, and default parameters were used for both. The classification accuracy is computed by averaging over ten trials of a 10-fold cross-validation experiment. Standard deviation is computed similarly.

5.2.1 Data Sets

Ten data sets were selected 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 [23]. In this data set, the target values are drugs' binding affinity to a particular protein. Five proteins are used to 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. 50 chemical structures were randomly sampled from the decoy list.

The next data set, from Wessel et al. [53] includes compounds classified by affinity for absorption through human intestinal lining. More over, the Predictive Toxicology Challenge [14] data sets were included, which contain a series of chemical compounds classified according to their toxicity in male rats, female rats, male mice, and female mice.

The same protocol was used as in [17] to transform chemical structure data set to graphs. Table 5.1 lists the total number of chemical compounds in each data set, as well as the number of positive and negative samples. In the table, # G: number of samples (chemical compounds) in the data set. # P: positive samples. # N: negative samples

Table 5.1. Data set and class statistics for GPD experiments.

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

5.2.2 Feature Sets

Frequent patterns were exclusively used from graph representations of chemicals in our study. Such frequent subgraphs were generated from a data set using two different graph mining approaches: that with exact matching [17] and that of approximate matching. In the approximate frequent subgraph mining, 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 the experiments approximate graph mining with $k = 1$ was used.

Once frequent subgraphs are mined, three feature sets are generated: (i) general subgraphs (all of mined subgraphs), (ii) tree subgraphs, and (iii) path subgraphs. Cycles were examined as well, not included 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.

5.2.3 Classification Methods

The performance of the following classifiers was evaluated.

- CBA. The first is a classifier that uses frequent itemset mining, known as Classification Based on Association (CBA) [3]. In CBA mined frequent subgraphs are treated as item sets.
- Graph Convolution Kernels. This type of kernel include the mismatch kernel (MIS) and the min-max (MNX) kernel. The former is based on the normalized Hamming distance of two binary vectors, and the latter is computed as the ratio between two sums: the numerator is the sum of the minimum between each feature pair in two binary vectors, and the denominator is the same except it sums the maximum. See [52] for details about the min-max kernel.
- SVM built-in Kernels. A linear kernel (Linear) and radial basis function (RBF) kernel was used.
- GPD. The graph pattern diffusion kernel was implemented as discussed in Section 5.1. The default parameter for the GPD kernel is a diffusion rate of λ =20% and the diffusion time $\tau = 5$.

5.2.4 Experimental Results

Here the results of our graph classification experiments are presented. One round of experiments was performed 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, patterns were selected that were general graphs, trees, and paths.

A simple feature selection method is applied in order to identify the most discriminating frequent patterns. Using a simple statistical formula, Pearson correlation coefficient (PCC), the correlation is measured 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% patterns are selected to construct the feature set.

5.2.4.1 Comparison between classifiers

The results of the comparison of different graph kernel functions are shown in Table 5.3. For this results, frequent subgraph mined using exact matching was used. In the table using general subgraphs (the first 10 rows in Table 5.3), it is shown that for exact mining of general subgraphs, in 4 of the 10 data sets, the GPD method provides mean accuracy that is significantly better (at least two standard deviations above the next best method). In another 4 data sets GPD gives the best performance, but the difference is less significant but is still more than 1 standard deviation). In the last two data sets other methods perform better, but not significantly better. The mismatch and min-max kernels all give roughly the same performance and hence only the results of the mismatch kernel are shown. The GPD's superiority is also confirmed in classifications where tree and path patterns are used.

Table 5.2 compares 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, the GDA method still gives the best performance for 6 of the seven data sets. These data sets were also tested using the recursive optimal-assignment kernel included in the JOELib2 computational chemistry library. It's results are comparable to those of the CBA method and hence were not included as separate results here.

Table 5.2. Comparison of GPD kernel to CBA.

Data set	GPD	CBA
CDK2 inhibitors	88.6*	80.46
COX2 inhibitors	82.7*	77.86
Fxa inhibitors	89.3*	86.87
PDE5 inhibitors	81.9	87.14*
A1A inhibitors	91.4*	87.76
intestinal absorption	63.14*	54.36
toxicity (male rats)	56.66*	55.95

In addition, a classifier called XRules was tested. XRules is designed for classification of tree data [60]. Chemical graphs, while not strictly trees, often are close to trees. To run the XRules executable, a graph is transformed to a tree by randomly selecting a spanning tree of the original graph. Our experimental study shows the application of XRules on average delivers incompetent results among the group of classifiers (e.g. 50% accuracy on the CDK2 inhibitor data set), which may be due to the particular way a graph is transformed to a tree. Since tree patterns are computed for the rule based classifier CBA in our comparison, XRules was not explored further.

A method based on a recursive optimal-assignment [10] was also tested 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, the chemical descriptors are ignored and the focus is instead on the structural alignment. In these experiments the performance of this method is very similar to CBA and hence the results of CBA only are shown.

5.2.4.2 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, approximate subgraph mining was also used to generate the features for our respective kernel methods. In both cases the general subgraphs mined are filtered into sets of trees and sets of paths as well.

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. These results are not recorded here since they add no appreciable information to the evaluation of the various methods.

The results using approximate subgraph mining (shown in Table 5.4) are similar to those for exact subgraph mining (shown in Table 5.3). In contrast to the hypothesis that using approximate subgraph mining might improve the classification accuracy, the data show that there is no significant difference between the set of features. However, it is clear that GPD is still better than the competing kernel functions.

Table 5.3. Comparison of different graph kernel functions and feature sets in GPD experiments, with strict subgraph matching.

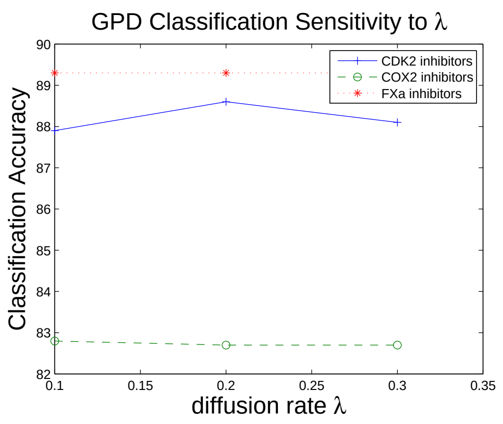
subgraph type	data set	MIS		GPD		Linear		RBF	
general	CDK2	76.3	2.06	87.2*	2.04	76.3	2.06	77.9	1.6
	COX2	85.1*	0.99	83.2	0.79	85.1*	0.99	84.5	1.08
	FXa	87	0.94	87.6*	0.52	87	0.94	86.2	0.42
	PDE5	83.2*	0.63	82.8	1.4	83.2*	0.63	83	0.67
	A1A	84.8	0.63	90.9*	0.74	85	0.94	88.7	1.06
	int. abs.	49.53	4.82	56.86*	3.12	50.7	4.56	47.56	3.44
	toxicity (FM)	51.46	3.4	54.81*	1.16	51.95	3.26	50.95	2.75
	toxicity (FR)	52.99	4.33	56.35*	1.13	49.57	4.71	51.94	3.34
	toxicity (MM)	49.64	3.43	60.71*	1.16	49.38	1.96	51.16	2.28
	toxicity (MR)	50.44	3.06	56.83*	1.17	49.91	3.09	54.3	2.59
trees	CDK2	76.3	2.06	87.2*	2.04	76.3	2.06	77.9	1.6
	COX2	85.1*	0.99	83.2	0.79	85.1*	0.99	84.5	1.08
	FXa	87	0.94	87.6*	0.52	87	0.94	86.2	0.42
	PDE5	83.2*	0.63	82.8	1.4	83.2*	0.63	83	0.67
	A1A	84.8	0.63	90.9*	0.74	85	0.94	88.7	1.06
	int. abs.	49.53	4.82	56.86*	3.12	50.7	4.56	47.56	3.44
	toxicity (FM)	51.46	3.4	54.81*	1.16	51.95	3.26	50.95	2.75
	toxicity (FR)	52.99	4.33	56.35*	1.13	49.57	4.71	51.94	3.34
	toxicity (MM)	49.64	3.43	60.71*	1.16	49.38	1.96	51.16	2.28
	toxicity (MR)	50.44	3.06	56.83*	1.17	49.91	3.09	54.3	2.59
paths	CDK2	76.3	0.82	86.2*	2.82	76.4	0.97	77.1	0.74
	COX2	85*	0	83.7	0.48	85*	0	85*	0
	FXa	86.8	0.79	87.6*	0.52	86.8	0.79	86.6	0.84
	PDE5	82.6	0.84	83*	1.25	82.6	0.84	82.7	0.95
	A1A	84.1	0.88	91.2*	1.14	84	0.67	85.7	0.67
	int. abs.	49.07	7.16	54.07*	3.52	50.58	4.32	50	4.72
	toxicity (FM)	50.14	3.41	54.79*	2.13	50.37	2.59	50.14	4.38
	toxicity (FR)	47.83	6.85	55.93*	2.44	48.32	7.83	50.09	4.37
	toxicity (MM)	46.85	3.57	58.81*	1.07	48.6	4.78	50.33	2.29
	toxicity (MR)	50.26	3.13	54.71*	1.38	48.69	3.93	54.27	3.04

Table 5.4. Comparison of different graph kernel functions and feature sets in GPD experiments, with approximate subgraph matching.

subgraph type	data set	MIS		GPD		Linear		RBF	
general	CDK2	76.3	2.06	85.7*	1.49	76.3	2.06	77.9	1.6
	COX2	85*	0	83	0.67	85*	0	85*	0
	FXa	86.4	0.52	87.5*	0.53	86.4	0.52	86.1	0.32
	PDE5	83.3*	0.67	83.3*	1.64	83.3*	0.67	82.9	0.74
	A1A	86.2	1.81	88.7*	0.82	86.2	1.81	88.7	0.48
	int. abs.	51.28	4.3	60.81*	2.63	52.67	4.07	51.86	6.18
trees	CDK2	76.3	2.06	85.7*	1.49	76.3	2.06	77.9	1.6
	COX2	85*	0	83	0.67	85*	0	85*	0
	FXa	86.4	0.52	87.5*	0.53	86.4	0.52	86.1	0.32
	PDE5	83.3*	0.67	83.3*	1.64	83.3*	0.67	82.9	0.74
	A1A	86.2	1.81	88.7*	0.82	86.2	1.81	88.7*	0.48
	int. abs.	51.28	4.3	60.81*	2.63	52.67	4.07	51.86	6.18
paths	CDK2	76.3	0.82	86.1*	2.13	76.4	0.97	77.1	0.74
	COX2	85*	0	83.4	0.7	85*	0	85*	0
	FXa	86	0	88*	0.82	86	0	86	0
	PDE5	83.1	0.57	83.8*	2.53	83.1	0.57	82.9	0.57
	A1A	83.6	0.7	88.6*	0.7	83.6	0.7	85.7	0.67
	int. abs.	49.88	4.3	60.23*	4.34	51.05	3.82	49.65	3.76

5.2.4.3 Effect Of Varying GPD Diffusion Rate And Time

This section evaluates the sensitivity of the GPD methods to its two parameters: diffusion rate λ and diffusion time. Different diffusion rate λ values and diffusion time values were tested. Figure 5.2 shows that the GPD algorithm is not very sensitive to the two parameters at the range that was tested. Although only three data sets are shown in Figure 5.2, the observation is true for other data sets in the experiments.



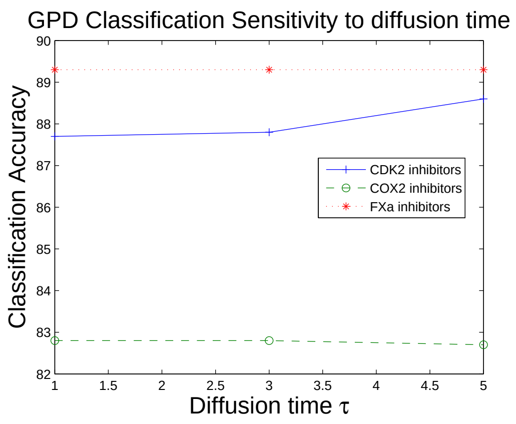


Figure 5.2. Effect of diffusion rate and time on GPD classification accuracy.

5.3 Conclusions

With the rapid development of fast and sophisticated data collection methods, data has 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. This work demonstrates the utility of a novel graph kernel function, graph pattern diffusion kernel (GPD kernel). It is shown 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 a very efficient computational framework was developed, computing a GPD kernel may be hard for large graphs. Future work will concentrate on improving the computational efficiency of the GPD kernel for very large graphs, as well as performing additional comparisons between this method other 2D-descriptor and QSAR-based methods.

Chapter 6

Matching Kernels with Approximate Pattern-based Features

This chapter expands on the GPD kernel presented in the previous chapter, by defining a similar kernel function that uses a matching-based set kernel instead of an alignment kernel. This method is termed a Graph Pattern Matching (GPM) kernel. The advantage of this modification is that the GPM kernel, unlike GPD, is guaranteed to be positive semi-definite, and hence a true Mercer kernel. This algorithm was tested using 16 chemical structure data sets. The experimental results demonstrate that this method outperforms existing state-of-the-art methods with a large margin.

6.1 GPM: A Graph Pattern Matching Kernel with Diffusion for Accurate Graph Classification

This section presents the design of a graph matching kernel with diffusion. The section begins by first presenting a general framework for graph matching. Then the pattern based graph matching kernel is presented. Finally a technique called "pattern diffusion" is discussed that significantly improves graph classification accuracy in practice.

6.1.1 Graph Matching Kernel

To derive an efficient algorithm scalable to large graphs, a function $\Gamma : V \rightarrow \mathbb{R}^n$ is used to map nodes in a graph to a n dimensional feature space that captures not only the node label information but also the neighborhood topological information around the node. If there is such a function Γ, the following graph kernel may be defined:

$$K_m(G, G') = \sum_{(u,v) \in V[G] \times V[G']} K(\Gamma(u), \Gamma(v)) \tag{6.1}$$

K can be any kernel function defined in the co-domain of Γ. This function K_m is called a *graph matching kernel*. The following theorem indicates that K_m is symmetric and positive semi-definite and hence a real kernel function.

Theorem 6.1.1 *The graph matching kernel is symmetric and positive semi-definite if the function K is symmetric and positive semi-definite.*

Proof sketch: the matching kernel is a special case of the R-convolution kernel and is hence positive semi-definite as proved in [31].

The kernel function can be visualized by constructing a weighted complete bipartite graph: connecting every node pair (u,v) $\in V[G] \times V[G']$ with an edge. The weight of the edge (u,v) is $K(\Gamma(v), \Gamma(v))$. Figure 6.1 shows a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$. Highlighted edges $(v1, u2)$, $(v2, u1)$, $(v3, u3)$ have larger weights than the rest of the edges (dashed).

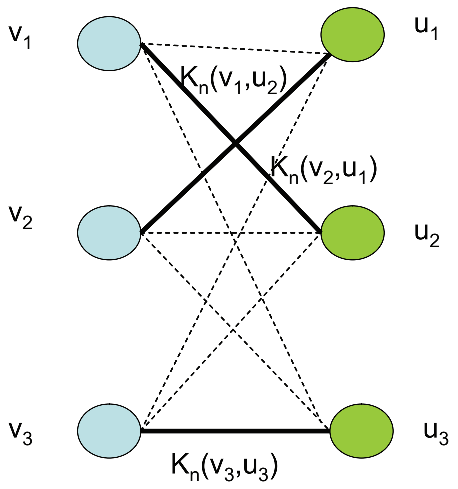


Figure 6.1. The maximum weighted bipartite graph for graph matching.

From the figure it can be seen that if two nodes are quite dissimilar, the weight of the related edge is small. Since dissimilar node pairs usually outnumber similar node pairs, if a linear kernel is used for nodes, kernel function may be noisy and hence lose the signal. In this design, the RBF kernel function is used, as specified below, to penalize dissimilar node pairs.

$$K(X, Y) = e^{\frac{-||X-Y||_2^2}{2}} \tag{6.2}$$

where $||X||_2^2$ is the squared L_2 norm of a vector X.

6.1.2 Graph Pattern Matching Kernel

One way to design the function Γ is to take advantage of frequent patterns mined from a set of graphs. Intuitively if a node belongs to a subgraph F, there is some information about the local topology of the node. Following the intuition, given a node v in a graph G and a frequent subgraph F, a function Γ_F is designed such that

$$\Gamma_F(v) = \begin{cases} 1 & \text{if } u \text{ belongs an embedding of } F \text{ in } G \\ 0 & \text{otherwise} \end{cases}$$

The function Γ_F is called a "pattern membership function" since this function tests whether a node occurs in a specific subgraph feature ("membership to a subgraph").

Given a set of frequent subgraphs $\mathcal{F} = F_1, F_2, \ldots, F_n$, each membership function is treated as a dimension and the function $\Gamma_{\mathcal{F}}$ is defined as below:

$$\Gamma_{\mathcal{F}}(v) = (\Gamma_{F_i}(v))_i^n \tag{6.3}$$

In other words, given n frequent subgraph, the function Γ maps a node v in G to a n-dimensional space, indexed by the n subgraphs, where values of the features indicate whether the node is part of the related subgraph in G.

Example 6.1.1 *In Figure 6.2, it is shown that two subgraph features F_1 and F_2. F_1 have an embedding in Q at $\{q_1, q_2\}$ and F_2 occurs in Q at $\{q_1, q_3\}$. The occurrences are depicted using shadings with different color and orientations. For node q_1, a subgraph F_1 is considered as a feature, and $\Gamma_{F_1}(q_1) = 1$ since q_1 is part of an embedding of F_1 in Q. Also, $\Gamma_{F_1}(q_3) = 0$ since q_3 is not part of an embedding* *of F_1 in Q. Similarly, $\Gamma_{F_2}(q_1) = 1$ and $\Gamma_{F_2}(q_3) = 1$. Hence $\Gamma_{F_1,F_2}(q_1) = (1,1)$ and $\Gamma_{F_1,F_2}(q_3) = (0,1)$. The values of the function Γ_{F_1,F_2} are also illustrated in the same figure using the annotated Q.*

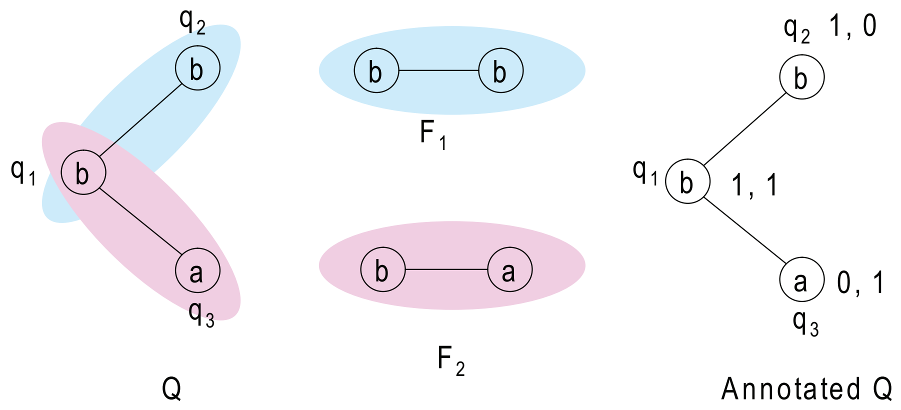


Figure 6.2. Example pattern membership functions for GPM kernel.

6.1.3 Graph Pattern Matching Kernel with Pattern Diffusion

This section introduces a better technique than the pattern membership function to capture the local topology information of nodes. This technique is called "pattern diffusion". It's design has the following advantages:

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

Below, the pattern diffusion kernel is outlined in three steps.

In the first step, a seed is identified as a starting point for the diffusion. In this design, a "seed" could be a single node, or a set of connected nodes in the original graph. In the experimental study, frequent subgraphs are always used for seeds since a seed from one graph can be easily compared to a seed in another graph.

In the second step given a set of nodes S as seed, a diffusion function f_t is recursively defined in the following way.

The base f_0 is defined as:

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

Define f_{t+1} ($t \geq 0$) with f_t in the following way:

$$f_{t+1}(v) = f_t(v) \times (1 - \frac{\lambda}{d(v)}) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)} \tag{6.4}$$

In the notation, $N(v) = \{u|(u, v) \text{ is an edge }\}$ is the set of nodes that connects to v directly. $d(v) = |N(v)|$ is the node degree of v. λ is a parameter that controls the diffusion rate.

The formula 6.4 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. It is called "diffusion" because the process simulate the way a value is spreading in a network. The 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, one parameter called diffusion time, denoted by τ, is used to control the diffusion process. Specifically the diffusion process is limited to a local region of the original graph with nodes that are at most τ hops away from a node in the seed S. In this sense, the diffusion should be named "local diffusion".

Finally in the last step, for the seed S, define the mapping function Γ^d_S as the limit function of f_t as t approaches to infinity, or

$$\Gamma^d_S = \lim_{t \to \infty} f_t \tag{6.5}$$

And given a set of frequent subgraph $\mathcal{F} = F_1, F_2, \ldots, F_n$ as seeds, define the pattern diffusion function $\Gamma^d_{\mathcal{F}}$ as:

$$\Gamma^d_{\mathcal{F}}(v) = (\Gamma^d_{F_i}(v))^n_i \tag{6.6}$$

6.1.4 Connections of Other Graph Kernels

6.1.4.1 Connection to Marginalized Kernels

Here the connection of pattern matching kernel function to the marginalized graph kernel [24] is shown, 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, 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, $U_{t+1} = M \times U_t$ where the matrix M is defined as:

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

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

Notice that the matrix M corresponds to a probability matrix corresponding to a Markov Chain since

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

Therefore the vector $M^\infty \times U_0$ corresponds to the stationary distribution of the local random walk as specified by M. In other words, rather than using random walk to retrieve information about the local topology of a graph, the stationary distribution is used to retrieve information about the local topology. The experimental study shows that this in fact is an efficient method of graph classification.

6.1.4.2 Connection to Optimal Assignment Kernel

The optimal assignment (OA) kernel [10] carries the same spirit of the graph pattern matching kernel in that OA uses pairwise node kernel function to construct a graph kernel function. OA kernel has been utilized for cheminformatics applications and is found to deliver good results empirically.

There are two major differences between GPM and the OA kernel. (1) OA kernel is not positive semi-definite and hence is not Mercer kernel in a strict sense. Non Mercer kernel functions are used to train SVM model and the problem is that the convex optimizer utilized in SVM will not converge to a global optimal and hence the performance of the SVM training may not be reliable. (2) OA utilizes a complicated recursive function to compute the similarity between nodes, which make the computation of the kernel function runs slowly for large graphs [45].

6.1.5 Pattern Diffusion Kernel and Graph Classification

This section summarizes the discussions presented so far and shows how the kernel function is utilized to construct an efficient graph classification algorithm in both the training and testing phases.

6.1.5.1 Training Phase

In the training phase, graphs of the training data set $D = \{(G_i, T_i,)\}_{i=1}^n$ are divided into groups according to their class labels. For example in binary classification, two groups of graphs: positive or negative. For multi-class classification, graphs are partitioned according to their class label where graphs have the same class labels are grouped together. The training phase is composed of four steps:

- Obtain frequent subgraphs. Identify frequent subgraphs from each graph group and union the subgraph sets together as the seed set $\mathcal{F}$.
- For each graph G in the training data set, use the node pattern diffusion function $\Gamma_{\mathcal{F}}^d$ to label nodes in G. Thus the feature vector of a node v is a vector $L_V = (\Gamma_{F_i}^d(v))_{i=1}^m$ with length $m = |\mathcal{F}|$.
- For two graphs G, G', construct the complete weighted bipartite graph as described in section 6.1.1 and compute the kernel $K_m(G, G')$ using Equation 6.1 and Equation 6.2.
- Train a predictive model using a kernel classifier.

6.1.5.2 Testing Phase

In the testing phase, the kernel function is computed for graphs in the testing and training data sets. The trained model is used to make predictions about graph in the testing set.

- For each graph G in the testing data set, use $\Gamma^d_{\mathcal{F}}$ to label nodes in G and create feature vectors as in the training phase.
- Use Equation 6.1 and Equation 6.2 to compute the kernel function $K_m(G, G')$ 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

The section below presents the empirical study of different kernel functions including the pattern matching kernel.

6.2 Experimental Study

Classification experiments were conducted using six different graph kernel functions, including the pattern diffusion kernel, on sixteen different data sets. There are twelve chemical-protein binding data sets, and the rest are chemical toxicity data sets. All of the experiments were performed on a desktop computer with a 3Ghz Pertium 4 processor and 1 GB of RAM. The following subsections describe the data sets and the classification methods in more detail along with the associated results.

In all classification experiments, the LibSVM [4] was used as kernel classifier. The nu-SVC was used with default parameter $\nu = 0.5$. The classification accuracy (TP+TN/S, TP: true positive, TN: true negative, S: total number of testing samples) is computed by averaging over a 10-fold cross-validation experiment.

Standard deviation is computed similarly. To have a fair comparison, default SVM parameters were used in all cases, and were not tuned to increase the accuracy of any method.

6.2.1 Data Sets

Sixteen data sets were selected, covering prediction of chemical-protein binding activity and chemical toxicity. The first seven data sets are manually extracted from the BindindDB database [30]. The next five are established data sets taken from Jorissen et al. [23]. The last four are from the Predictive Toxicology Challenge [14] (PTC). Detailed information for the data sets is available in the following table, where # G: number of samples (chemical compounds) in the data set. # P: positive samples. # N: negative samples .

Table 6.1. Characteristics of data sets in GPM experiments.

Source	Dataset	# G	# P	# N
BindingDB	AChE	138	69	69
	ALF	93	47	46
	EGF-R	377	190	187
	HIV-P	202	101	101
	HIV-RT	365	183	182
	HSP90	82	41	41
	MAPK	255	126	129
Jorissen	CDK2	100	50	50
	COX2	100	50	50
	FXa	100	50	50
	PDE5	100	50	50
	A1A	100	50	50
Predictive Toxicology Challenge	PTC-FM	344	152	192
	PTC-FR	336	129	207
	PTC-MM	351	121	230
	PTC-MR	349	143	206

6.2.1.1 BindingDB Sets

The BindingDB database contains more than 450 proteins. For each protein, the database record chemicals that bind to the protein. Two types of activity measurements K_i and IC_{50} are provided. Both measurements measure inhibition/dissociation rates between a proteins and chemicals. From BindingDB, 7 proteins were manually selected with a wide range of known interacting chemicals (ranging from tens to several hundreds). These data sets are AChE, ALF, EGF-R, HIV-P, HIV-RT, HSP90, and MAPK.

6.2.1.2 Jorissen Sets

The Jorissen data sets also contains information about chemical-protein binding activity. In this case the provider of the data set carefully selected positive and negative samples and hence are more reliable than the data sets created from BindingDB. For more information about the creation of the data sets, see [23] in details. The data sets from this study are: CDK2, COX2, FXa, PDE5, and A1A.

6.2.1.3 PTC Sets

The Predictive Toxicology Challenge (PTC) data sets contain a series of chemical compounds classified according to their toxicity in male rats, female rats, male mice, and female mice. While chemical-protein binding activity is an important type of chemical activity, it is not the only type. Toxicity is another important, though different, kind of chemical activity necessary to predict in drug design. These data sets (PTC-FR/FM/MR/MM) are well curated and highly reliable.

6.2.2 Kernel Functions

Six different kernel functions were selected for evaluation: Marginalized [24], spectrum [7], tanimoto [29], subtree [33], optimal assignment [10], together with the graph pattern matching kernel.

Four kernel functions (Marginalized, spectrum, tanimoto, subtree) are computed using the open source Chemcpp v1.0.2 package [36]. The optimal assignment kernel was computed using the JOELib2 package, and is not strictly a kernel function, but still provides good prediction accuracy. The graph pattern matching kernel was computed using MATLAB code.

6.2.3 Experimental Results

6.2.3.1 Comparison Between Kernel Functions

This subsection presents the results of our graph classification experiments with various kernel functions. Figure 6.3 shows the classification accuracy for different kernel functions and data sets, averaged over a 10-fold cross validation experiment. The standard deviations (omitted) of the accuracies are generally very high, from 5-10%, so statistically significant differences between kernel functions are generally not observed.

The data shows that the GPM method is competitive for all sixteen data sets. If the accuracy of each kernel function averaged over all data sets is examined, the GPM kernel performs the best overall. Again, the standard deviations are high so the differences between the top performing kernels are not statistically significant. Still, with 16 different data sets some trends are clear: GPM kernel delivers the highest classification accuracy in 8 out of the 16 data sets, with tanimoto kernel best in 4, marginalized best in 2, subtree in 2, optimal assignment in 1 and

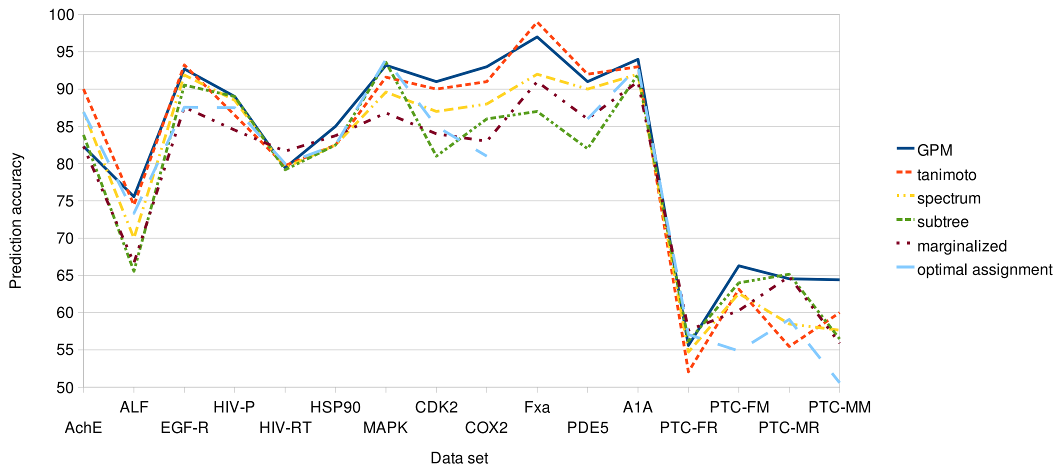


Figure 6.3. Average accuracy for kernel functions and data sets in GPM experiments.

spectrum in none.

Although GPM does not work well on a few data sets such as AChE, HIV-RT, MAPK, and PTC-FR/MR, overall it performs better when compared to any other kernel for a majority of data sets. It is better than every other kernel function in at least 10 of the 16 data sets.

In general the GPM, spectrum and tanimoto kernels perform the best, with over all average accuracy of about 80%. The subtree, optimal assignment, and marginalized also perform very good, in mid to high 70%. The min/max tanimoto kernel performed much worse than the other methods, and hence it was not included in the figure. Note that the optimal assignment kernel is missing a prediction accuracy for the FXa data set, this was due to a terminal error in the JOELib2 software used to calculate this kernel on this data set.

6.3 Conclusions

Graphs have proven to be powerful tools for modeling complex, high-dimensional biological data; building highly accurate predictive models for chemical graph classification is a goal for cheminformatics and drug design. This work demonstrates the utility of a novel graph kernel function, graph pattern matching kernel (GPM kernel). It is shown that the GPM kernel can capture the intrinsic connection between a chemical and its class label and has the lowest testing error in majority of the data sets we evaluated.

Chapter 7

Graph Wavelets for Topology Comparison

Previous kernels such as alignment other substructure-based kernels attempt to mitigate the high-dimensionality of graphs in different ways. The first possibility is to use complex patterns, such as general subgraphs, but restrict pattern selection in some way. The second approach is to use simpler patterns such as paths or trees, but retain the set of feature patterns. In the most extreme case, graphs are reduced to point sets of vertices for very fast but information-poor analysis. The approach presented here, termed Wavelet-Alignment (WA) kernel, works on simpler graph representations but uses an application of graph wavelet analysis to create high-quality localized structure features for chemical analysis. The wavelet functions are used to condense neighborhood information about an atom into a single feature of that atom, rather than features spread over it's neighboring atoms. By doing so, (local) features are extracted with various topological scales about chemical structures and use these wavelet features to compute an alignment of two chemical graphs. This chapter describes the wavelet-alignment method in detail and compares it to competing methods for chemical activity prediction with several data sets.

7.1 Graph Wavelet Alignment Kernels for Drug Virtual Screening

The following sections outline the algorithms that drive our experimental method. This method measures the similarity of graph structures whose nodes and edges have been labeled with various features. These features represent different kinds of chemical structure information including atoms and chemical bonds types among others. To compute the similarity of two graphs, the nodes of one graph are aligned with the nodes of the second graph, such that the total overall similarity is maximized with respect to all possible alignments. Vertex similarity is measured by comparing vertex descriptors, and is computed recursively so that when comparing two nodes, the immediate neighbors of those nodes are also compared, and the neighbors of those neighbors, and so on.

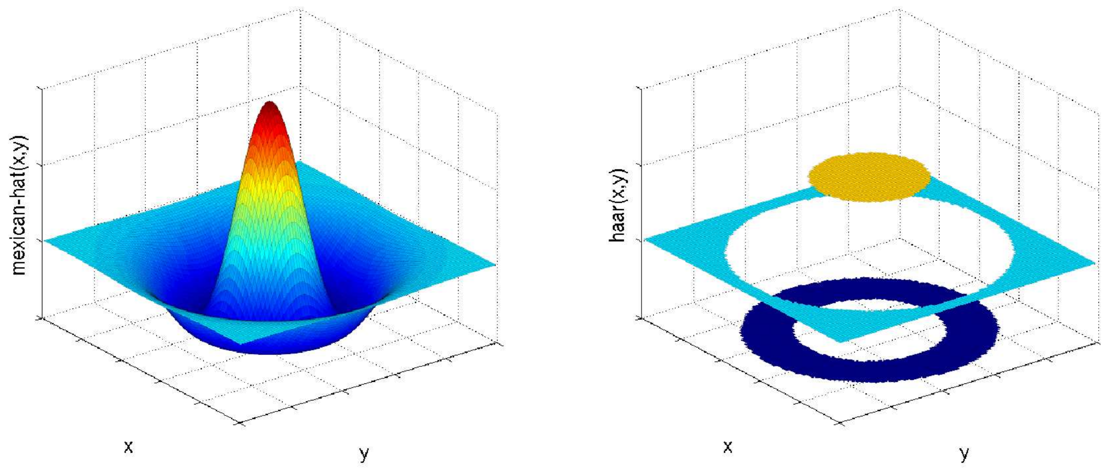


Figure 7.1. Two wavelet functions in three dimensions, Mexican hat and Haar.

7.1.1 Graph Alignment Kernel

An *alignment* of two graphs G and G' (assuming $|V[G] \leq |V[G']|$) is a 1-1 mapping $\pi : V[G] \rightarrow V[G']$. Given an alignment π, define the similarity between two graphs, as measured by a kernel function k_A, below:

$$\begin{aligned} k_A(G, G') := \max_\pi \textstyle\sum_{v \in V[G]} k_n(v, \pi(v)) + \\ \textstyle\sum_{u,v} k_e((u, v), (\pi(u), \pi(v))) \end{aligned} \tag{7.1}$$

The function k_n is a kernel function to measure the similarity of nodes and the function k_e is a kernel function to measure the similarity of edges. Intuitively, equation 7.1 use an additive model to compute the similarity between two graphs by computing the sum of the similarity of nodes and the similarity of edges. The maximal similarity among all possible alignments is defined as the similarity between two graphs.

7.1.2 Simplified Graph Alignment Kernel

A direct computation of the graph alignment kernel is computationally intensive and is unlikely scalable to large graphs. With no surprise, the graph alignment kernel computation is no easier than the subgraph isomorphism problem, a known NP-hard problem [1]. To derive efficient algorithms scalable to large graphs, the graph kernel function is simplified with the following formula:

$$k_M(G, G') = \max_\pi \sum_{v \in V[G]} k_a(f(v), f(\pi(v))) \tag{7.2}$$

[1]Formally, showing a reduction from the graph alignment kernel to the subgraph isomorphism problem is needed. The details of such reduction are omitted due to their loose connection to the main theme of the current paper, which is advanced data mining approach as applied to cheminformatics applications

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 that not only include node features but also include information about topology of the graph where v belongs to.

Equation 7.2, computes a maximal weighted bipartite graph, which has an efficient solution known as the Hungarian algorithm. The complexities of the algorithm is $O(|V[G]|^3)$. See [10] for further details.

Provided below is an efficient method, based on graph wavelet analysis, to create features to capture the topological structure of a graph.

7.1.3 Graph Wavelet Analysis

Originally proposed to analyze time series signals, wavelet analysis transforms a series of signals to a set of summaries with different scale. Two of the key insights of wavelet analysis of signals are (i) using localized basis functions and (ii) analysis with different scales. Wavelet analysis offers efficient tools to decompose and represent a function with arbitrary shape [6, 11]. Since invented, wavelet analysis has quickly gained popularity in a wide range of applications outside time series data, such as image analysis and geography data analysis. In all these applications, the level of detail, or *scale* is considered as an important factor in data comparison and compression. Figure 7.1 shows two examples of wavelet functions in a 3D space, the Haar and Mexican Hat.

Intuition. With wavelet analysis as applied to graph representations chemical structure, for each atom, features about the atom and its local environment are collected at different scales. For example, information can be collected about the average charge of an atom and it's surrounding atoms, then assign the average value as a feature to the atom. Information can also be collected about whether an atom belongs to a nearby functional group, whether the surrounding atoms of a particular atom belong to a nearby functional group, and the local topology of an atom to its nearby functional groups.

In summary, conceptually the following two types of insights are gained about the chemicals after applying wavelet analysis to graph represented chemical structure:

- *Analysis with varying levels of scale.* Intuitively, at the finest level, two chemical structures are compared by matching the atoms and chemical bonds in the two structures. At the next level, comparison of two regions is performed (e.g. chemical functional groups). At an even coarser level, small regions may be grouped into larger ones (e.g. pharmacophore), and two chemicals are compared by matching the large regions and the connections among large regions.

- *Non-local connection.* In a chemical structure, two atoms that are not directly connected by a chemical bond may still have some kind of interaction. Therefore when comparing two graphs and their vertices cannot depend only on the *local environment* immediately surrounding an atom, but rather must consider both local and non-local environment.

Though conceptually appealing, current wavelet analysis is often limited to numerical data with regularly structures such as matrices and images. Graphs, however, are arbitrarily structured and may represent innumerable relationships and topologies between data elements. In order to define a reasonable graph wavelet functions, the following two important concepts are introduced:

- h-hop neighborhood

- Discrete wavelet functions

The former, h-hop neighborhood, is essentially used to project graphs from a high dimensional space with arbitrary topology into a Euclidean space suitable for operation with wavelets. The h-hop measure defines a distance metric between vertices that is based on the shortest path between them. The discrete wavelet function then operates on a graph projection in the h-hop Euclidean space to compactly represent the information about the local topology of a graph. It is the use of this compact wavelet representation in vertex comparison that underlies the complexity reduction achieved by this method. Based on the h-hop neighborhood, a discrete wavelet function is used to summarize information in a local region of a graph and create features based on the summarization. These two concepts are discussed in detail below.

7.1.3.1 h-hop neighborhood

In this section the following definitions are introduced.

Definition 7.1.1 *Given a node v in a graph G the **h-hope neighborhood** of v, denoted by $N_h(v)$, is the set of nodes that are (according to the shortest path) exactly h hops away from v.*

For example if $h = 0$, then $N_0(v) = v$ and if $h = 1$, then $N_1(v) = \{u|(u, v) \in E[G]\}$.

Here f_v denotes the feature vector associated with a node v in a graph G. $|f|$ is the feature vector length (number of features in the feature vector). The average feature measurement, denoted by $\overline{f}_j(v)$ for nodes in $N_j(v)$ is

$$\overline{f}_j(v) = \frac{1}{|N_j(v)|} \sum_{u \in N_j(v)} f_u \tag{7.3}$$

Example 7.1.1 *The left part of the Figure 7.2 shows a chemical graph. Given a node v in the graph G, label the shortest distance of nodes to v in the G. In this case $N_0(v) = v$ and $N_1(v) = \{t, u\}$. If the feature vector contains a single feature of atomic number, $\overline{f}_1(v)$ is the average atomic number of atoms that are at most 1-hop away from v. In this case, since $N_1(v) = \{t, u\}$ and $\{t, u\}$ are both carbon with atomic number equal to eight, then $\overline{f}_1(v)$ is equal to eight as well.*

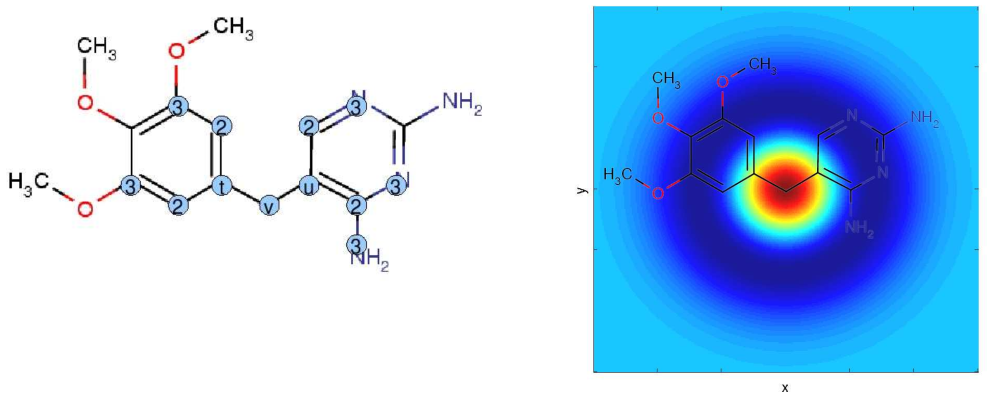


Figure 7.2. A chemical graph and hop distances.

7.1.3.2 Discrete wavelet functions

In order to adapt a wavelet function to discrete structure such as graphs, a wavelet function $\psi(x)$ must be applied to the h-hop neighborhood. Towards that end, a wavelet function $\psi(x)$ (such as the Haar, or Mexican Hat) can be scaled to have support on the domain $[0, 1)$, with integral 0, and partition the function into $h+1$ intervals. Then compute the average, $\psi_{j,h}$, as the average of $\psi(x)$ over the jth interval, $0 \leq j \leq h$ as below.

$$\psi_{j,h} \equiv \frac{1}{h+1} \int_{j/(h+1)}^{(j+1)/(h+1)} \psi(x) dx \tag{7.4}$$

With neighborhood and discrete wavelet functions, wavelet analysis can be applied to graphs. This analysis is called *wavelet measurements*, denoted by $\Gamma_h(v)$, for a node v in a graph G at scale up to $h > 0$.

$$\Gamma_h(v) = C_{h,v} * \sum_{j=0}^{h} \psi_{j,h} * \overline{f}_j(v) \tag{7.5}$$

where $C_{h,v}$ is a normalization factor with $C(h,v) = (\sum_{j=0}^{h} \frac{\psi_{j,h}^2}{|N_k(v)|})^{-1/2}$

Define $\Gamma^h(v)$ as the sequence of wavelet measurements as applied to a node v with scale value up to h. That is $\Gamma^h(v) = \{\Gamma_1(v), \Gamma_2(v), \ldots, \Gamma_h(v)\}$. Call $\Gamma^h(v)$ the *wavelet measurement vector* of node v. Finally insert the wavelet measurement vector into the alignment kernel with the following formula.

$$k_\Gamma(G, G') = \max_\pi \sum_{v \in V[G]} k_a(\Gamma^h(v), \Gamma^h(\pi(v))) \tag{7.6}$$

where $k_a(\Gamma^h(v), \Gamma^h(\pi(v))$ is a kernel function defined on vectors. Two popular choices are linear kernel and radius based function kernel.

Example 7.1.2 *The right part of Figure 7.2 shows a chemical graph overlayed with a wavelet function centered on a specific vertex. It is clear how the wavelet is most intense at the central vertex, hop distance of zero, corresponding to a strongly positive region of the wavelet function. As the hop distance increases* *the wavelet function becomes strongly negative, roughly at hop distances of one and two. At hop distance greater than two, the wavelet function returns to zero intensity, indicating negligible contribution from vertices at this distance.*

7.2 Experimental Study

Classification experiments were conducted on five different biological activity data sets, and measured support vector machine (SVM) classifier prediction accuracy for several different feature generation methods. The following sections describe the data sets and classification methods in more detail, along with the associated results.

We performed all of experiments on a desktop computer with a 3Ghz Pertium 4 processor and 1 GB of RAM.

7.2.1 Data Sets

Five data sets were selected to represent typical chemical benchmarks in drug design to evaluate the classifier performance. The Predictive Toxicology Challenge data set, discussed by Helma et al [14], contains a set of chemical compounds classified according to their toxicity in male rats (PTC-MR), female rats (PTC-FR), male mice (PTC-MM), and female mice (PTC-FM). The Human Intestinal Absorption (HIA) data set (Wessel et al. [53]) contains chemical compounds classified by intestinal absorption activity. The remaining data set (MD) is from Patterson et al [35], and was used to validate certain molecule descriptors. Various statistics for these data sets can be found in Table 7.1.

All of these data sets exist natively as binary classification problems, therefore in the case of the MD and HIA data sets, some preprocessing is required to

Table 7.1. Data set and class statistics for WA experiments.

Dataset	# Graphs	Class	Labels	Count
HIA	86	regression	0 - 100	86
		binary	0	39
			1	47
		multi-class	1	21
			2	18
			3	21
			4	26
MD	310	regression	0 - 7000	310
		binary	0	162
			1	148
		multi-class	1	46
			2	32
			3	37
			4	35
PTC-MR	344	binary	0	192
			1	152
PTC-MM	336	binary	0	207
			1	129
PTC-FR	351	binary	0	230
			1	121
PTC-FM	349	binary	0	206
			1	143

transform them into regression and multi-class problems. For regression, this is a straightforward process of using the compound activity directly as the regression target. In the case of multi-class problems the transformation is not as direct. A histogram of compound activity values was chosen to visualize which areas of the activity space are more dense, allowing natural and intuitive placement of class separation thresholds.

7.2.2 Methods

The performance of the SVM classifier trained with different methods was evaluated. The first two methods (WA-linear, WA-RBF) are both computed using the wavelet-alignment kernel, but use different functions for computing atom-atom similarity; both a linear and RBF function were tested here. Different hop distance thresholds were evaluated and fixed to $h = 3$ in all experiments.

The method optimal alignment (OA) consists of the similarity values computed by the optimal assignment kernel, as proposed by Frölich et al [10]. There are several reasons that we consider OA as the current-state-of-the-art graph based chemical structure classification method. First, the OA method is developed specifically for chemical graph classification. Second the OA method contains a large library to compute different features for chemical structures. Third, the OA method has developed a sophisticated kernel function to compute the similarity between two chemical structures. The experimental study shows that the wavelet analysis obtains performance profiles comparable to, and sometimes exceeding that of the existing state-of-the-art chemical classification approaches. In addition, a significant computation time reduction was achieved by using the wavelet analysis. The details of the experimental study are shown below.

In these experiments, the support vector machine (SVM) classifier was used in order to generate activity predictions. The LibSVM classifier was used, as implemented by Chang et al [4] and included in the Weka data-mining software package by Witten et al. [56]. The SVM parameters were fixed across all methods, and we use a linear kernel. For (binary) classification nu-SVC was used for multi-class classification with nu = 0.5. The Haar wavelet function was used in the WA experiments. Classifier performance was averaged over a 10-fold cross-validation set.

Most of the algorithms were developed and tested under the MATLAB programming environment. The OA software was provided by [10] as part of their JOELib software, a computational chemistry library implemented in java. [17]

7.2.3 Results

Below are results of the experimental study of the wavelet-alignment kernel with two focuses: (i) classification accuracy and (ii) computational efficiency.

7.2.3.1 Classification Accuracy

Table 7.2. Prediction results of cross-validation trials for WA experiments.

Dataset	Labels	OA	WA-RBF	WA-linear
HIA	real	979.82(32.48)*	989.72(33.60)	989.31(24.62)
	binary	51.86(3.73)	61.39(2.77)*	57.67(3.54)
	multi-class	29.30(2.23)	39.06(0.63)*	29.76(5.73)
MD	real	3436395(1280)	3436214(1209)*	3440415(1510)
	binary	67.16(0.86)*	52.51(3.34)	65.41(0.42)
	multi-class	39.54(1.65)*	33.35(3.83)	33.93(1.87)
PTC-FM	binary	58.56(1.53)*	51.46(3.45)	55.81(1.31)
PTC-FR	binary	58.57(2.11)	52.87(2.65)	59.31(1.95)*
PTC-MM	binary	58.23(1.25)	52.36(0.93)	58.91(2.078)*
PTC-MR	binary	51.51(1.20)	52.38(3.48)	52.09(2.61)*

Table 7.2.3.1 reports the average and standard deviation of the prediction results over 10 trials. The best results are marked with an asterisk. For classification problems, results are in prediction accuracy, and for regression problems they are in mean squared error (MSE) per sample. From the table, observe that for the HIA data set, WA-RBF kernel significantly outperforms OA for both binary and multi-class classification. For MD data set, OA does best for both classification

sets, but WA-linear is best for regression. For the PTC binary data, the WA-linear method outperforms the others in 3 of the 4 sets.

7.2.3.2 Computational Efficiency

In Table 7.2.3.2, the kernel computation time for both OA and WA methods was documented using 6 different data sets. The runtime advantage of the WA algorithm over OA is clear, showing improved computation efficiency by factors of over 10 fold for the WA-linear kernel and over 5 fold for the WA-RBF kernel.

Figure 7.2.3.2 shows the kernel computation time across a range of dataset sizes, with chemical compounds sampled from the HIA data set. Using simple random sampling with replacement, data sets were created sized from 50 to 500. OA method was not run on even larger data sets since the experimental results clearly demonstrate the efficiency of the WA kernel already.

What these run time results do not demonstrate is the *even greater* computational efficiency afforded by the WA algorithm when operating on general, non-chemical graph data. As noted at the end of section four, chemical graphs have some restrictions on their general structure. Specifically, the number of atom neighbors is bound by a small constant (4 or so). Since the OA computation time is much more dependent on the number of neighbors, WA is even more advantageous in these circumstances. Unfortunately, since the OA software is designed as part of the JOELib chemoinformatics library specifically for use with chemical graphs, it will not accept generalized graphs as input, and hence this aspect of the algorithm could not be empirically demonstrated

Table 7.3. Running time results for WA experiments.

Dataset	Kernel	Time	Speedup
HIA	OA	75.87	-
	WA-RBF	13.76	5.51
	WA-linear	4.91	15.45
MD	OA	350.58	-
	WA-RBF	50.85	6.89
	WA-linear	26.82	13.07
PTC-FM	OA	633.13	-
	WA-RBF	103.95	6.09
	WA-linear	44.87	14.11
PTC-FR	OA	665.95	-
	WA-RBF	116.89	5.68
	WA-linear	54.64	12.17
PTC-MM	OA	550.41	-
	WA-RBF	99.39	5.53
	WA-linear	47.51	11.57
PTC-MR	OA	586.12	-
	WA-RBF	101.68	5.80
	WA-linear	45.93	12.73

7.3 Conclusions

Graph structures are a powerful and expressive representation for chemical compounds. This work presents a new method *wavelet-assignment*, for computing the similarity of chemical compounds, based on the use of an optimal assignment graph kernel function augmented with pattern and wavelet based descriptors. The experimental study demonstrates that this wavelet-based method delivers an improved classification model, along with an order of magnitude speedup in kernel computation time. For high-volume, real world data sets, this algorithm is able to handle a much greater number of graph objects, demonstrating it's potential for processing both chemical and non-chemical data in large amounts. In the present study, only a limited number of atom features are used. In the future, domain

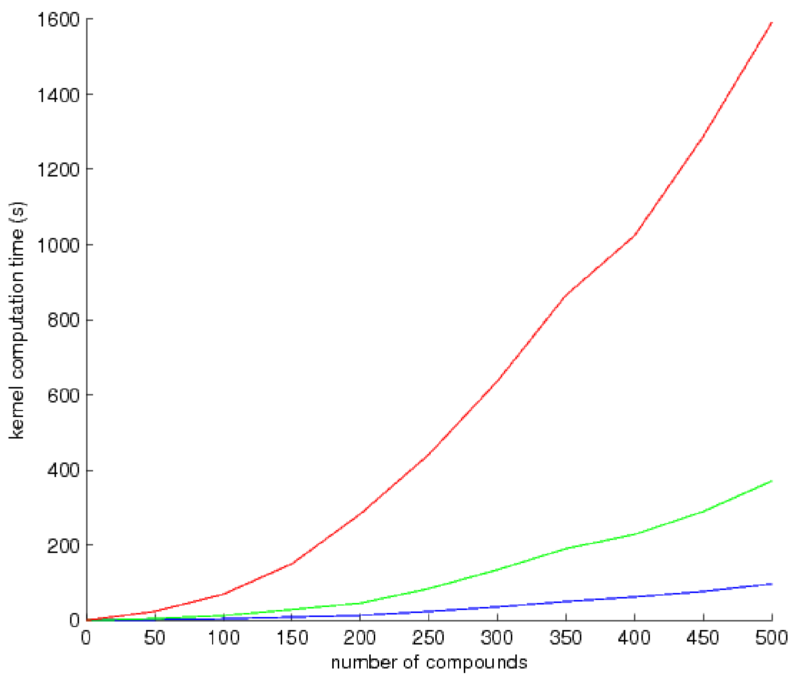


Figure 7.3. Comparison of computation times between methods for WA experiments.

experts can be involved to evaluate the performance of these algorithms, including the prediction accuracy and the capability for identifying important features in diverse chemical structure data sets.

Chapter 8

Future Work and Overall Conclusions

Graph structures are a powerful and expressive representation for many kinds of data. With the rapid development of fast and sophisticated data collection methods, data has become complex, high-dimensional and noisy. Graphs have proven to be powerful tools for modeling such data. Building highly accurate predictive models for graph data is a new challenge for the data mining and machine learning communities. These methods are of great value in a variety of fields but especially so in biological and chemical research where computational tools are helping to make many important new discoveries with respect to disease treatment and other medical activities.

Much recent activity on graph classification has focused on the definition of kernel functions for comparing graphs objects directly. The kernel function defines an implicit feature space where graph classification can be accomplished via support vector machine or other kernel classifier. Classification in kernel space avoids many difficulties associated with using high-dimensional feature vectors to represent graphs and other complex objects.

The use of kernel functions do not completely mitigate the problems of working with complex graph objects, however. Currently established kernel functions are either slow to compute or lack discriminative power. This thesis addresses these issues through several novel techniques, however there remain many opportunities for further improvement. In the chemical domain, at least, there appear to be many high level structural rules that even complex models have difficulty capturing. The most precise models are prohibitively time consuming for databases of the size now available.

Future work *must* focus on methods for efficient, large scale analysis. The value of this high volume approach is exemplified by the proliferation of high-throughput screening technology, which has drastically accelerated the analysis of chemicals and biological molecules. The ability to accurately and quickly analyze databases in the millions of compounds using only computer models offers unprecedented opportunities for learning about biological systems.

Ultimately, the result of improved computer models is better understanding and control of complex phenomena. Biological systems, though an important beneficiaries of such models, are only a single area of potential application. Graphs are fundamental to our general understanding of many concepts. Therefore, only by fully understanding graphs can these concepts themselves be fully modeled and understood.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. Network flows. *SIAM Review*, 37 No.1, 1995.

[2] CP Austin, LS Brady, TR Insel, and FS Collins. Nih molecular libraries initiative. *Science*, 306(5699):1138–9, 2004.

[3] Yiming Ma Bing Liu, Wynne Hsu. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

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

[5] M. Crovella and E. Kolaczyk. Graph wavelets for spatial traffic analysis. *Infocom*, 3:1848–1857, 2003.

[6] Antonios Deligiannakis and Nick Roussopoulos. Extended wavelets for multiple measures. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

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

[8] F. Fontaine, M. Pastor, I. Zamora, and F. Sanz. Anchor-grind: Filling the gap between standard 3d qsar and the grid-independent descriptors. *J. Med. Chem.*, 48:2687–2694, 2005.

[9] H. Fröhlich, J. Wegner, F. Sieker, and A. Zell. Kernel functions for attriubted molecular graphs - a new similarity-based approach to adme prediction in classification. *QSAR & Combinatorial Science*, 25(4):317–326, 2006.

[10] Fröohlich, J. Wegner, F. Sieker, and A. Zell. Kernel functions for attributed molecular graphs - a new similarity-based approach to adme prediction in classification. *QSAR & Combinatorial Science*, 2006.

[11] M. Garofalakis and Amit Kumar. Wavelet synopses for general error metrics. *ACM Transactions on Database Systems (TODS)*, 30(4):888–928, 2005.

[12] 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.

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

[14] C. Helma, R. King, and S. Kramer. The predictive toxicology challenge 2000-2001. *Bioinformatics*, 17(1):107–108, 2001.

[15] Tamas Horvath, Jan Ramon, and Stefan Wrobel. Frequent subgraph mining in outerplanar graphs. In *SIGKDD*, 2006.

[16] 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.

[17] Jun Huan, Wei Wang, and Jan 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.

[18] Jun Huan, Wei Wang, Jang Prins, and Jiong Yang. SPIN: Mining maximal frequent subgraphs from graph databases. pages 581–586, 2004.

[19] Yu Huang, Haifeng Li, Haiyan Hu, Xifeng Yan, Michael S. Waterman, Haiyan Huang, and Xianghong Jasmine Zhou. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, pages ISMB/ECCB Supplement, 222–229, 2007.

[20] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. *In PKDD'00*, pages 13–23, 2000.

[21] W. Wang J. Huan and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. *In Proc. of ICDM*, 2003.

[22] Ruoming Jin, Scott Mccalle, , and Eivind Almaas. Trend motif: A graph mining approach for analysis of dynamic complex networks. *ICDM*, 2007.

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

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

[25] Yiping Ke, James Cheng, , and Wilfred Ng. Correlation search in graph databases. In *SIGKDD*, 2007.

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

[27] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. An application of boosting to graph classification. In *NIPS*, 2004.

[28] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. International Conference on Data Mining'01*, pages 313–320, 2001.

[29] Ralaivola L, Swamidass SJ, Saigo H, and Baldi P. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, 2005.

[30] T. Liu, Y. Lin, X. Wen, R. N. Jorrisen, and M.K. Gilson. Bindingdb: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research*, 35:D198–D201, 2007.

[31] Siwei Lyu. Mercer kernels for object recognition with local features. In *IEEE Computer Vision and Pattern Recognition*, pages 223–229, 2005.

[32] M. Maggioni, J. Bremer Jr, R. Coifman, and A. Szlam. Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. In *Proc. SPIE Wavelet XI*, volume 5914, 2005.

[33] P. Mahe and J.P. Vert. Graph kernels based on tree patterns for molecules. Technical Report HAL:ccsd-00095488, Ecoles des Mines de Paris, September 2006.

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

[35] D. Patterson, R. Cramer, A. Ferguson, R. Clark, and L. Weinberger. Neighbourhood behaviour: A useful concept for validation of "molecular diversity" descriptors. *J. Med. Chem.*, 39:3049–3059, 1996.

[36] Jean-Luc Perret, Pierre Mahe, and Jean-Philippe Vert. Chemcpp: an open source c++ toolbox for kernel functions on chemical compounds, 2007. Software available at http://chemcpp.sourceforge.net.

[37] J. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization.* MIT Press, Cambridge, MA, 1998.

[38] J. Ross Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kaufmann, 1993.

[39] Put R, Xu QS, Massart DL, and Vander Heyden Y. Multivariate adaptive regression splines (mars) in chromatographic quantitative structure-retention relationship studies. *J Chromatogr A.*, 1055(1-2), 2004.

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

[41] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceeding of the ACM Symposium on Principles of Database Systems (PODS)*, 2002.

[42] Aaron Smalter, Jun Huan, and Gerald Lushington. Gpd: A graph pattern diffusion kernel for accurate graph classification. In *Proceedings of the 8th International Workshop on Data Mining in Bioinformatics*, 2008.

[43] Aaron Smalter, Jun Huan, and Gerald Lushington. Gpm: A graph pattern matching kernel with diffusion for accurate graph classification. In *Proceedings of the 8th IEEE International Conference on BioInformatics and Bio-Engineering*, 2008.

[44] Aaron Smalter, Jun Huan, and Gerald Lushington. Graph wavelet alignment kernels for drug virtual screening. In *Proceedings of the 7th Annual International Conference on Computational Systems Bioinformatics*, 2008.

[45] Aaron Smalter, Jun Huan, and Gerald Lushington. Structure-based pattern mining for chemical compound classification. In *Proceedings of the 6th Asia Pacific Bioinformatics Conference*, 2008.

[46] Jimeng Sun, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Parameter-free mining of large time-evolving graphs. In *SIGKDD*, 2007.

[47] V. Svetnik, C. Tong A. Liaw, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: A classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43, 2003.

[48] Stefan Wrobel Tamas Horvath, Thomas Gartner. Cyclic pattern kernels for predictive graph mining. *SIGKDD*, 2004.

[49] Nicola Tolliday, Paul A. Clemons, Paul Ferraiolo, Angela N. Koehler, Timothy A. Lewis, Xiaohua Li, Stuart L. Schreiber, Daniela S. Gerhard, and Scott

Eliasof. Small molecules, big players: the national cancer institute's initiative for chemical genetics. *Cancer Research*, 66:8935–42, 2006.

[50] Hanghang Tong, Yehuda Koren, , and Christos Faloutsos. Fast direction-aware proximity for graph mining. In *SIGKDD*, 2007.

[51] V. Vapnik. *Statistical Learning Theory.* John Wiley, 1998.

[52] Nikil Wale, Ian Watson, , and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 2007.

[53] M. Wessel, P. Jurs, J. Tolan, and S. Muskal. Prediction of human intestinal absorption of drug compounds from molecular structure. *J. Chem. Inf. Comput. Sci.*, 38(4):726–735, 1998.

[54] Jason Weston, Rui Kuang, Christina Leslie, and William Stafford Noble. Protein ranking by semi-supervised network propagation. *BMC Bioinformatics*, 2006.

[55] David Williams, Jun Huan, and Wei Wang. Graph database indexing using structured graph decomposition. In *in Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*, 2007.

[56] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, CA, 2nd edition edition, 2005.

[57] Y. Xue, H. Li, C. Y. Ung, C. W. Yap, and Y. Z. Chen. Classification of a diverse set of tetrahymena pyriformis toxicity chemical compounds from molecular descriptors by statistical learning methods. *Chem. Res. Toxicol.*, 19 (8), 2006.

[58] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proc. International Conference on Data Mining'02*, pages 721–724, 2002.

[59] X. Yan, P. S. Yu, and J. Han. Graph indexing based on discriminative frequent structure analysis. In *ACM Transactions on Database Systems (TODS)*, 2005.

[60] Mohammed J. Zaki and Charu C. Aggarwal. Xrules: An effective structural classifier for xml data. *Machine Learning Journal special issue on Statistical Relational Learning and Multi-Relational Data Mining*, 62, No. 1-2:137–170, 2006.

[61] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *SIGKDD*, 2006.

[62] D. Zhou, J. Huang, and B. Schöolkopf. Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.