GPM: A Graph Pattern Matching Kernel with Diffusion for Chemical Compound Classification

Aaron Smalter, Jun Huan and Gerald Lushington

***Abstract*— Classifying chemical compounds is an active topic in drug design and other cheminformatics applications. Graphs are general tools for organizing information from heterogenous sources and have been applied in modelling many kinds of biological data. With the fast accumulation of chemical structure data, building highly accurate predictive models for chemical graphs emerges as a new challenge .**

In this paper, we demonstrate a novel technique called Graph Pattern Matching kernel (GPM). Our idea is to leverage existing frequent pattern discovery methods and explore their application to kernel classifiers (e.g. support vector machine) for graph classification. In our method, we first identify all frequent patterns from a graph database. We then map subgraphs to graphs in the database and use a diffusion process to label nodes in the graphs. Finally the kernel is computed using a set matching algorithm. We performed experiments on 16 chemical structure data sets and have compared our methods to other major graph kernels. The experimental results demonstrate excellent performance of our method.

I. INTRODUCTION

The fast accumulation of data describing chemical structures [1] 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[25].

Publicly-available large-scale chemical compound databases have offered tremendous opportunities for creating highly efficient *in silico* drug design methods. Many machine learning and data mining algorithms have been applied to study the structure-activity relationship of chemicals with the goal of building classifiers for graph-structured data. Additional applications include protein function prediction based on structure [9] and gene regulation networks analysis [10].

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 and several have been studied.

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.

Aaron Smalter and Jun Huan are with Department of Electrical Engineering and Computer Science, University of Kansas, USA asmalter,jhuan@ku.edu

Gerald Lushington is with the Molecular Graphics and Modeling Laboratory, University of Kansas, USA glushington@ku.edu

The initial work was done by Haussler in his work of *R-convolution* kernel, providing a framework of which many current graph kernel function follow [7]. Recent progress of 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 [6], random walk based kernels [12], and kernels based on shortest paths between pair of nodes [13]. The second group of kernel functions try to capture the local similarity of two graphs by counting the shared subcomponents of graphs. These include the subtree kernels [20], cyclic kernels [24], spectrum kernel [4], and recently subgraph kernels [23].

In this paper, we explore the second avenue and aim to leverage existing frequent pattern mining algorithms in building accurate graph kernel functions. Towards that end, we demonstrate a novel technique called graph pattern matching kernel (GPM). We have tested our algorithm using 16 chemical structure data sets. The experimental results demonstrate that our method outperforms existing state-of-the-art methods with a large margin.

The rest of the paper is organized as follows. In the remainder of this section, we give a brief survey of research efforts that are closely related to our current work. In section II, we provide background information about graphs. In section III, we present the details of our graph pattern matching kernels. In section IV we use real-world data sets to evaluate our proposed methods and perform a comparison of ours to the current state-of-the-art. Finally we conclude and present our future plan in section V.

A. Related Work

We survey the work related to graph classification methods by dividing them into two categories. The first category of methods explicitly collect a set of *features* from the graphs. Once a set of features is determined, a graph is described by a feature vector, and any existing classification methods such as Classification based on Association (CBA) [2] and decision tree [19] that work in an *n*-dimensional Euclidian space, may be applied for graph classification.

The second approach is to implicitly collect a (possibly infinite) set of features from graphs. Rather than computing the features, this approach computes the similarity of graphs, using the framework of “kernel functions” [26]. The advantage of a kernel method is that it has low chance of over fitting, which is a serious concern in high dimensional space with low sample size. We review these kernel functions in the following section.

1) Kernel Functions for Graphs: In recent years a variety of graph kernel functions have been developed, with promising application results as described by Ralaviola *et al.* [21].

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 [6]. 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, Kashima *et al.* [12] developed the *marginalized kernel* that uses a Markov model to generate random walks of a labeled graph. The kernel is then computed using the number of shared walks.

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 [4]. Transformations of the inner product, such as min-max kernel [27] and Tanimoto kernel [14], are also widely used. The subtree kernel [17] is a variation on the spectrum kernel that uses subtrees instead of paths.

The optimal assignment kernel, proposed by Frölich et al [5], differs significantly from the marginalized graph kernel in that it attempts to align two graphs, rather than compare sets of linear substructures. The similarity between the two graphs is computed by finding the maximal weighted bipartite graph between the two sets of nodes.

II. BACKGROUND

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

Definition 2.1: 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: $V \cup E \to \Sigma$ is a function that assigns labels to vertices and edges. We assume that a total ordering is defined on the labels in Σ.

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

Definition 2.2: 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 $f : V' \to V$ 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. We say 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 2.1: Figure 1 shows a graph database of three labeled graphs. The mapping (isomorphism) $q_1 \to p_3$, $q_2 \to p_1$, and $q_3 \to p_2$ demonstrates that graph Q is subgraph

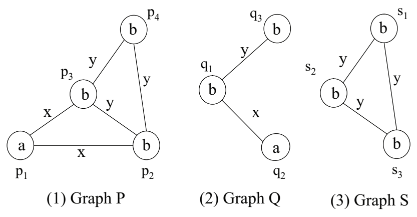

Fig. 1. A Database of three labeled graphs.

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.

III. GRAPH PATTERN MATCHING KERNELS

Here we present our design of a graph matching kernel with diffusion. We start the section by first presenting a general framework for graph matching. Then we present the pattern based graph matching kernel. Finally we show a technique we call "pattern diffusion" that significantly improves graph classification accuracy in practice.

A. Graph Matching Kernel

To derive an efficient algorithm scalable to large graphs, our idea is to use a function $\Gamma : V \to \mathbb{R}^n$ 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 we have such function Γ, we may design the following graph kernel:

$$K_m(G, G') = \sum_{(u,v) \in V[G] \times V[G']} K(\Gamma(u), \Gamma(v)) \qquad (1)$$

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

Theorem 3.1: 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 [16].

We visualize the kernel function 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 $K(\Gamma(v), \Gamma(v))$. In Figure 2, we show a weighted complete bipartite graph for $V[G] = \{v_1, v_2, v_3\}$ and $V[G'] = \{u_1, u_2, u_3\}$.

From the figure we see 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 we use linear kernel for nodes, we may have a noisy kernel function and hence loose our signal. In our design, we use the RBF kernel function, as specified below, to penalize dissimilar node pairs.

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

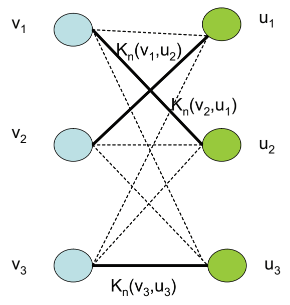

Fig. 2. The maximum weighted bipartite graph for graph matching. Highlighted edges $(v1,u2)$, $(v2,u1)$, $(v3,u3)$ have larger weights than the rest of the edges (dashed).

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

B. 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, we have 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, we design a function Γ_F such that

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

We call the function Γ_F as 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 subgraph $\mathscr{F} = F_1, F_2, \ldots, F_n$, we treat each membership function as a dimension and design the function $\Gamma_{\mathscr{F}}$ as below:

$$\Gamma_{\mathscr{F}}(v) = (\Gamma_{F_i}(v))_i^n \tag{3}$$

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 3.1: In Figure 3, we duplicated the figure Q in Figure 1. We show two subgraph features F_1 and F_2. F_1 has an embedding in Q at $\{q_1, q_2\}$ and F_2 occurs in Q at $\{q_1, q_3\}$. We depict the occurrences using shadings with different color and orientations. For node q_1, if we consider subgraph F_1 as a feature, we have $\Gamma_{F_1}(q_1) = 1$ since q_1 is part of an embedding of F_1 in Q. Also, we have $\Gamma_{F_1}(q_3) = 0$ since q_3 is not part of an embedding of F_1 in Q. Similarly we have $\Gamma_{F_2}(q_1) = 1$ 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.

C. Graph Pattern Matching Kernel with Pattern Diffusion

Here we introduce a better technique than the pattern membership function to capture the local topology information of nodes. We call this technique "pattern diffusion". Our design has the following advantages:

- Our design is generic and does not assume any domain knowledge from a specific application. The diffusion process may be applied to graphs with dramatically different characteristics.
- The diffusion process is straightforward to implement and can be computed efficiently.
- We prove that the diffusion process is related to the probability distribution of a graph random walk. This explains why the simple process may be used to summarize local topological information.

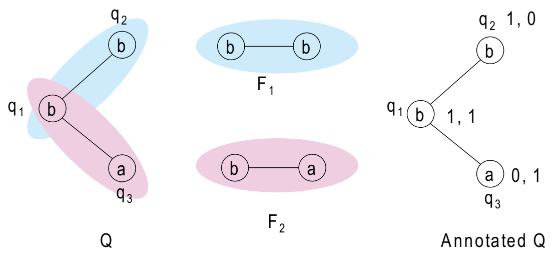

Fig. 3. An example of pattern membership functions.

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

In the first step, we identify a seed as a starting point for the diffusion. In our design, a "seed" could be a single node, or a set of connected nodes in the original graph. In our experimental study, we always use frequent subgraphs for seeds since we can easily compare a seed from one graph to a seed in another graph.

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

The base f_0 is defined as:

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

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

$$f_{t+1}(v) = f_t(v) \times (1 - \frac{\lambda}{d(v)}) + \sum_{u \in N(v)} f_t(u) \times \frac{\lambda}{d(u)} \tag{4}$$

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 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. We call it "diffusion" because the process simulate the way a value is spreading in a network. Our intuition is that the distribution of such a value encodes information about the local topology of the network.

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

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

$$\Gamma_S^d = \lim_{t\to\infty} f_t \tag{5}$$

And given a set of frequent subgraph $\mathscr{F} = F_1, F_2, \ldots, F_n$ as seeds, we design the pattern diffusion function $\Gamma_{\mathscr{F}}^d$ as:

$$\Gamma_{\mathscr{F}}^d(v) = (\Gamma_{F_i}^d(v))_i^n \tag{6}$$

D. Connections of Other Graph Kernels

1) Connection to Marginalized Kernels: Here we show the connection of pattern matching kernel function to the marginalized graph kernel [12], which uses a Markov model to randomly generate walks of a labeled graph.

Given a graph G with nodes set $V[G] = \{v_1, v_2, \ldots, v_n\}$, and a seed $S \subseteq V[G]$, for each diffusion function f_t, we construct a vector $U_t = (f_t(v_1), f_t(v_2), \ldots, f_t(v_n))$. According to the definition of f_t, we have $U_{t+1} = M \times U_t$ 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, we compute the stationary distribution ($f_S = \lim_{t\to\infty} f_t$) by computing $M^\infty \times U_0$.

We 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, we use the stationary distribution to retrieve information about the local topology. Our experimental study shows that this in fact is an efficient way for graph classification.

2) Connection to Optimal Assignment Kernel: The optimal assignment (OA) kernel [5] carries the same spirit of our 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 ours 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 [23].

E. Pattern Diffusion Kernel and Graph Classification

We summarize the discussions we present so far and show how the kernel function is utilized to construct an efficient graph classification algorithm at both the training and testing phases.

1) Training Phase: In the training phase, we divide graphs of the training data set $D = \{(G_i, T_i,)\}_{i=1}^n$ into groups according to their class labels. For example in binary classification, we have two groups of graphs: positive or negative. For multi-class classification, we partition graphs 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. We identify frequent subgraphs from each graph group and union the subgraph sets together as our seed set $\mathscr{F}$.
- For each graph G in the training data set, we use the node pattern diffusion function $\Gamma_{\mathscr{F}}^d$ 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 $m = |\mathscr{F}|$.
- For two graphs G, G', we construct the complete weighted bipartite graph as described in section III-A and compute the kernel $K_m(G, G')$ using Equation 1 and Equation 2.
- Train a predictive model using a kernel classifier.

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

- For each graph G in the testing data set, we use $\Gamma_{\mathscr{F}}^d$ to label nodes in G and create feature vectors as we did in the training phase.
- We use Equation 1 and Equation 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

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

IV. EXPERIMENTAL STUDY

We conducted classification experiments using six different graph kernel functions, including our Pattern Diffusion kernel, on sixteen different data sets. Due to space constraints we are unable to present the entirety of our data here, but it can be viewed in our technical report [22]. There are twelve chemical-protein binding data sets, and the rest are chemical toxicity data sets. We performed all of our experiments on a desktop computer with a 3Ghz Pertium 4 processor and 1 GB of RAM. In the following subsections, we describe the data sets and the classification methods in more detail along with the associated results.

In all classification experiments, we used the LibSVM [3] as our kernel classifier. We used nu-SVC with $\nu = 0.5$. Our classification accuracy (TP+TN/S, TP: true positive, TN: true negative, S: total number of testing samples) is computed

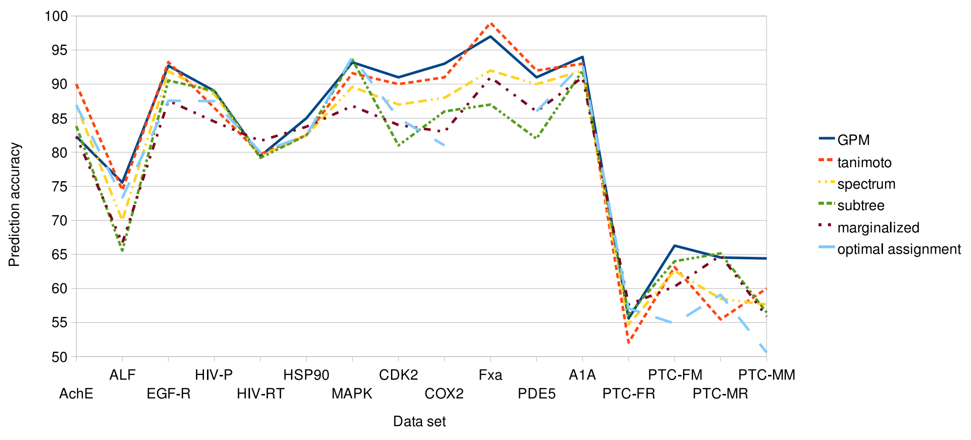

Fig. 4. Average accuracy for all kernel functions and data sets.

by averaging over a 10-fold cross-validation experiment. Standard deviation is computed similarly. To have a fair comparison, we simply used default SVM parameters in all cases, and did not tune any parameters to increase the accuracy of any method.

A. Data Sets

We have selected sixteen data sets covering prediction of chemical-protein binding activity and chemical toxicity. The first seven data sets are manually extracted from the BindindDB database [15]. The next five are established data sets taken from Jorissen et al. [11]. The last four are from the Predictive Toxicology Challenge[8] (PTC). Detailed information for the data sets is available in supplementary material.

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, we manually selected 7 proteins 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.

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 we created from BindingDB. For more information about the creation of the data sets, see [11] in details. The data sets from this study are: CDK2, COX2, FXa, PDE5, and A1A.

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 we would like to predict in drug design. These data sets (PTC-FR/FM/MR/MM) are well curated and highly reliable.

B. Kernel Functions

We have selected 6 different kernel functions for evaluation: Marginalized[12], spectrum[4], tanimoto[14], subtree[17], optimal assignment[5], together with our graph pattern matching kernel.

Four kernel functions (Marginalized, spectrum, tanimoto, subtree) are computed using the open source Chemcpp v1.0.2 package [18]. The optimal assignment kernel was computed using the JOELib2 package, and is not strictly a kernel function, but still provides good prediction accuracy. Our graph pattern matching kernel was computed using our own MATLAB code.

C. Experimental Results

1) Comparison Between Kernel Functions: Here we present the results of our graph classification experiments with various kernel functions. Figure 4 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.

We can see from the data that our method is competitive for all sixteen data sets. If we examine the accuracy of each kernel function averaged over all data sets, we see that our 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 we can see some clear trends: 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 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.

V. 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. In this paper we have demonstrated the utility of a novel graph kernel function, graph pattern matching kernel (GPM kernel). We showed 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.

REFERENCES

[1] C. Austin, L. Brady, T. Insel, and F. Collins. Nih molecular libraries initiative. *Science*, 306(5699):1138–9, 2004.

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

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

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

[5] 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.

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

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

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

[9] 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.

[10] Y. Huang, H. Li, H. Hu, X. Yan, M. S. Waterman, H. Huang, and X. J. Zhou. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, pages ISMB/ECCB Supplement, 222–229, 2007.

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

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

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

[14] R. L, S. SJ, S. H, and B. P. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, 2005.

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

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

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

[18] J.-L. Perret, P. Mahe, and J.-P. Vert. Chemcpp: an open source c++ toolbox for kernel functions on chemical compounds, 2007. Software available at http://chemcpp.sourceforge.net.

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

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

[21] L. Ravaliola, S. J. Swamidass, and H. Saigo. Graph kernels for chemical informatics. *Neural Networks*, 2005.

[22] A. Smalter, J. Huan, and G. Lushington. Gpm: A graph pattern matching kernel with diffusion for accurate graph classification. Technical report, University of Kansas, August 2008.

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

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

[25] N. Tolliday, P. A. Clemons, P. Ferraiolo, A. N. Koehler, T. A. Lewis, X. Li, S. L. Schreiber, D. S. Gerhard, and S. Eliasof. Small molecules, big players: the national cancer institute's initiative for chemical genetics. *Cancer Research*, 66:8935–42, 2006.

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

[27] N. Wale, I. Watson, , and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 2007.