Marginalized Kernels Between Labeled Graphs

Hisashi Kashima HKASHIMA@JP.IBM.COM
IBM Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, 242-8502 Kanagawa, Japan

Koji Tsuda KOJI.TSUDA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany; and
AIST Computational Biology Research Center, 2-43 Aomi, Koto-ku, 135-0064 Tokyo, Japan

Akihiro Inokuchi INOKUCHI@JP.IBM.COM
IBM Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, 242-8502 Kanagawa, Japan

Abstract

A new kernel function between two labeled graphs is presented. Feature vectors are defined as the counts of label paths produced by random walks on graphs. The kernel computation finally boils down to obtaining the stationary state of a discrete-time linear system, thus is efficiently performed by solving simultaneous linear equations. Our kernel is based on an infinite dimensional feature space, so it is fundamentally different from other string or tree kernels based on dynamic programming. We will present promising empirical results in classification of chemical compounds.[1]

1. Introduction

A large amount of the research in machine learning is concerned with classification and regression for real-valued vectors (Vapnik, 1998). However, much of the real world data is represented not as vectors, but as *graphs* including sequences and trees, for example, biological sequences (Durbin et al., 1998), natural language texts (Manning & Schütze, 1999), semi-structured data such as HTML and XML (Abiteboul et al., 2000), and so on. Especially, in the pharmaceutical area, the chemical compounds are represented as labeled graphs, and their automatic classification is of crucial importance in the rationalization of drug discovery processes to predict the effectiveness or toxicity of drugs from their chemical structures (Kramer & De Raedt, 2001; Inokuchi et al., 2000).

Kernel methods such as support vector machines are becoming increasingly popular for their high performance (Schölkopf & Smola, 2002). In kernel methods, all computations are done via a *kernel function*, which is the inner product of two vectors in a feature space. In order to apply kernel methods to graph classification, we first need to define a kernel function between the graphs. However, defining a kernel function is not an easy task, because it must be designed to be *positive semidefinite*[2]. Following the pioneering work by Haussler (1999), a number of kernels were proposed for structured data, for example, Watkins (2000), Jaakkola et al. (2000), Leslie et al. (2003), Lodhi et al. (2002), and Tsuda et al. (2002) for sequences, and Vishwanathan and Smola (2003), Collins and Duffy (2001), and Kashima and Koyanagi (2002) for trees. Most of them are based on the idea of an object decomposed into substructures (i.e. subsequences, subtrees or subgraphs) and a feature vector is composed of the counts of the substructures. As the dimensionality of feature vectors is typically very high, they deliberately avoid explicit computations of feature values, and adopt efficient procedures such as dynamic programming or suffix trees.

In this paper, we will construct a kernel function between two graphs, which is distinguished from the kernel between two vertices in a graph, e.g., diffusion kernels (Kondor & Lafferty, 2002; Kandola et al., 2003; Lafferty & Lebanon, 2003) or the kernel between two paths in a graph, e.g., path kernels (Takimoto & Warmuth, 2002). There has been almost no significant works for designing kernels between two graphs except

[1]An extended abstract of this research has been presented in IEEE ICDM International Workshop on Active Mining at Maebashi, Japan (Kashima & Inokuchi, 2002). But this full paper contains much richer theoretical analysises such as interpretation as marginalized kernels, convergence conditions, relationship to linear systems, and so on.

[2]Ad hoc similarity functions are not always positive semidefinite, e.g. Shimodaira et al. (2002) and Bahlmann et al. (2002).

for the work by Gärtner (2002). We will discuss about his kernels in Appendix.

One existing way to describe a labeled graph as a feature vector is to count *label paths* appearing in the graph. For example, the pattern discovery algorithm (Kramer & De Raedt, 2001; Inokuchi et al., 2000) explicitly constructs a feature vector from the counts of frequently appearing label paths in a graph. For the labeled graph shown in Figure 1, a label path is produced by traversing the vertices, and looks like

$$(A, e, A, d, D, a, B, c, D),$$

where the vertex labels A, B, C, D and the edge labels a, b, c, d, e appear alternately. Numerous label paths are produced by traversing the nodes in every possible way. Especially when the graph has a loop, the dimensionality of the count vector is *infinite*, because traversing may never end. In order to explicitly construct the feature vector, we have to select features to keep the dimensionality finite. The label paths may be selected simply by limiting the path length or, more intelligently, the pattern discovery algorithm identifies the label paths which appear frequently in the training set of graphs. In any case, there is an additional unwanted parameter (i.e. a threshold on path length or path frequency) that has to be determined by a model selection criterion (e.g. the cross validation error).

We will propose a kernel function based on the inner product between infinite dimensional path count vectors. A label path is produced by random walks on graphs, and thus is regarded as a random variable. Our kernel is defined as the inner product of the count vectors averaged over all possible label paths, which is regarded as a special case of *marginalized kernels* (Tsuda et al., 2002). The kernel computation boils down to finding the stationary state of a discrete-time linear system (Rugh, 1995), which can be done efficiently by solving simultaneous linear equations with a sparse coefficient matrix. We especially notice that this computational trick is fundamentally different from the dynamic programming techniques adopted in other kernels (e.g. Watkins (2000), Lodhi et al. (2002), Collins and Duffy (2001), Kashima and Koyanagi (2002)). These kernels deal with very large but still *finite* dimensional feature spaces and have parameters to constrain the dimensionality (e.g. the maximum length of subsequences in Lodhi et al. (2002)). In order to investigate how our kernel performs well on the real data, we will show promising results on predicting the properties of chemical compounds.

This paper is organized as follows. In Section 2, we review marginalized kernels as the theoretical foundation. Then, a new kernel between graphs is presented in Section 3. In Section 4, we summarize the results of our experiments on the classification of chemical compounds. Finally, we conclude with Section 5.

2. Marginalized Kernels

A common way for constructing a kernel for structured data such as strings, trees, and graphs is to assume hidden variables and make use of the probability distribution of visible and hidden variables (Haussler, 1999; Watkins, 2000; Tsuda et al., 2002). For example, Watkins (2000) proposed conditional symmetric independence (CSI) kernels which are described as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h}} p(\boldsymbol{x}|\boldsymbol{h}) p(\boldsymbol{x}'|\boldsymbol{h}) p(\boldsymbol{h}),$$

where $\boldsymbol{h}$ is a hidden variable, and $\boldsymbol{x}$ and $\boldsymbol{x}'$ are visible variables which correspond to structured data. This kernel requires $p(\boldsymbol{x}|\boldsymbol{h})$, which means that the generation process of $\boldsymbol{x}$ from $\boldsymbol{h}$ needs to be known. However, it may be the case that $p(\boldsymbol{h}|\boldsymbol{x})$ is known instead of $p(\boldsymbol{x}|\boldsymbol{h})$. Then we can use a *marginalized kernel* (Tsuda et al., 2002) which is described as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} K_z(\boldsymbol{z}, \boldsymbol{z}') p(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{h}'|\boldsymbol{x}'). \qquad (1)$$

where $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{h}]$ and $K_z(\boldsymbol{z}, \boldsymbol{z}')$ is the *joint kernel* depending on both visible and hidden variables. The posterior probability $p(\boldsymbol{h}|\boldsymbol{x})$ can be interpreted as a feature extractor that extracts informative features for classification from $\boldsymbol{x}$. The marginalized kernel (1) is defined as the expectation of the joint kernel over all possible values of $\boldsymbol{h}$ and $\boldsymbol{h}'$. In the following, we will construct a graph kernel in the context of the marginalized kernel. The hidden variable is a sequence of vertex indices, which is generated by random walks on the graph. Also the joint kernel K_z is defined as a kernel between the sequences of vertex and edge labels traversed in the random walk.

3. Graph Kernel

In this section, we introduce a new kernel between graphs with vertex labels and edge labels. At the beginning, let us formally define a labeled graph. Denote by G a labeled directed graph and by $|G|$ the number of vertices. Each vertex of the graph is uniquely indexed from 1 to $|G|$. Let $v_i \in \Sigma_v$ denote the label of vertex i and $e_{ij} \in \Sigma_E$ denote the label of the edge from i to j. Figure 1 shows an example of the graphs that we handle in this paper. We assume that there are no multiple edges between any vertices. Our task is to construct a kernel function $K(G, G')$ between two labeled graphs G and G'.

3.1. Random Walks on Graphs

The hidden variable $\boldsymbol{h} = (h_1, \ldots, h_\ell)$ associated with graph G is a sequence of natural numbers from 1 to $|G|$. Given a graph G, $\boldsymbol{h}$ is generated by a random walk as follows: At the first step, h_1 is sampled from the

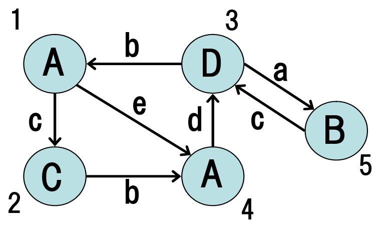

Figure 1. an example of graphs with labels

initial probability distribution $p_s(h)$. Subsequently, at the i-th step, the next vertex h_i is sampled subject to the transition probability $p_t(h_i|h_{i-1})$, but the random walk may also end with probability $p_q(h_{i-1})$:

$$\sum_{j=1}^{|G|} p_t(j|i) + p_q(i) = 1. \tag{2}$$

The posterior probability for the label path $\boldsymbol{h}$ is described as

$$p(\boldsymbol{h}|G) = p_s(h_1)\prod_{i=2}^{\ell} p_t(h_i|h_{i-1})p_q(h_\ell),$$

where ℓ is the length of $\boldsymbol{h}$.
When we do not have any prior knowledge, we cab set p_s to be the uniform distribution, the transition probability p_t to be an uniform distribution over the vertices adjacent to the current vertex, and the termination probability p_q to be a constant. Also gaps (Lodhi et al., 2002) can be incorporated by setting the transition probabilities appropriately.

3.2. Joint Kernel

Next we define the joint kernel K_z assuming that the hidden sequences $\boldsymbol{h}$ and $\boldsymbol{h}'$ are given for two graphs G and G', respectively. When the random walk is done as described in $\boldsymbol{h}$, the traversed labels are listed as

$$v_{h_1} e_{h_1 h_2} v_{h_2} e_{h_2 h_3} v_{h_3} \cdots$$

Assume that two kernel functions, $K(v, v')$ and $K(e, e')$, are readily defined between vertex labels and edge labels, respectively. We constrain both kernels $K(v, v'), K(e, e') \geq 0$ to be nonnegative[3]. An example of the vertex label kernels is

$$K(v, v') = \delta(v = v'), \tag{3}$$

where δ is a function that returns 1 if its argument holds, 0 otherwise. If the labels are defined in $\mathbb{R}$, the Gaussian kernel

$$K(v, v') = \exp(- \parallel v - v' \parallel^2 / 2\sigma^2) \tag{4}$$

[3]This constraint will play an important role in proving the convergence of the marginalized kernel in Section 3.4.

would be a natural choice (Schölkopf & Smola, 2002). The joint kernel is defined as the product of the label kernels as follows:

$$K_z(\boldsymbol{z}, \boldsymbol{z}') = \begin{cases} 0 & (\ell \neq \ell') \\ K(v_{h_1}, v'_{h'_1}) \prod_{i=2}^{\ell} K(e_{h_{i-1}h_i}, e'_{h'_{i-1}h'_i}) \times & \\ \qquad K(v_{h_i}, v'_{h'_i}) & (\ell = \ell') \end{cases},$$

where $\boldsymbol{z} = (G, \boldsymbol{h})$.

3.3. Efficient Computation

The marginalized kernel (1) is the expectation of K_z over all possible $\boldsymbol{h}$ and $\boldsymbol{h}'$, which is described as

$$\begin{aligned} &K(G, G') \\ &= \sum_{\ell=1}^{\infty} \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} p_s(h_1) \prod_{i=2}^{\ell} p_t(h_i|h_{i-1}) p_q(h_l) \times \\ &\qquad p'_s(h'_1) \prod_{j=2}^{\ell} p'_t(h'_j|h'_{j-1}) p'_q(h'_\ell) \times \\ &\qquad K(v_{h_1}, v'_{h'_1}) \prod_{k=2}^{\ell} K(e_{h_{k-1}h_k}, e'_{h'_{k-1},h'_k}) K(v_{h_k}, v'_{h'_k}), \end{aligned}$$

where $\sum_{\boldsymbol{h}} := \sum_{h_1=1}^{|G|} \cdots \sum_{h_\ell=1}^{|G|}$. The straightforward enumeration is obviously impossible, because ℓ spans from 1 to infinity. Nevertheless we have an efficient way to compute this kernel as shown below. By rearranging the terms, the kernel has the following nested structure.

$$\begin{aligned} K(G, G') = \lim_{L\to\infty} \sum_{\ell=1}^{L} \sum_{h_1, h'_1} s(h_1, h'_1) \times \qquad (5) \\ \left(\sum_{h_2, h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3, h'_3} t(h_3, h'_3, h_2, h'_2) \times \right.\right. \\ \left.\left. \left(\cdots \left(\sum_{h_\ell, h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) q(h_\ell, h'_\ell) \right) \right) \cdots \right) \end{aligned}$$

where

$$\begin{aligned} s(h_1, h'_1) &:= p_s(h_1) p'_s(h'_1) K(v_{h_1}, v'_{h'_1}) \\ t(h_i, h'_i, h_{i-1}, h'_{i-1}) &:= p_t(h_i|h_{i-1}) p'_t(h'_i|h'_{i-1}) \times \\ &\qquad K(v_{h_i}, v'_{h'_i}) K(e_{h_{i-1}h_i}, e_{h'_{i-1}h'_i}) \\ q(h_\ell, h'_\ell) &:= p_q(h_\ell) p'_q(h'_\ell) \end{aligned}$$

Let us further simplify (5) as

$$K(G, G') = \lim_{L\to\infty} \sum_{\ell=1}^{L} \sum_{h_1, h'_1} s(h_1, h'_1) r_\ell(h_1, h'_1),$$

where for $\ell \geq 2$,

$$r_\ell(h_1, h'_1) := \left(\sum_{h_2,h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3,h'_3} t(h_3, h'_3, h_2, h'_2) \times \left(\cdots \left(\sum_{h_\ell,h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) q(h_\ell, h'_\ell) \right) \right) \cdots \right) \right),$$

and $r_1(h_1, h'_1) := q(h_1, h'_1)$. Replacing the order of summation, we have the following:

$$\begin{aligned} K(G, G') &= \sum_{h_1,h'_1} s(h_1, h'_1) \lim_{L\to\infty} \sum_{\ell=1}^{L} r_\ell(h_1, h'_1) \\ &= \sum_{h_1,h'_1} s(h_1, h'_1) \lim_{L\to\infty} R_L(h_1, h'_1), \quad (6) \end{aligned}$$

where

$$R_L(h_1, h'_1) := \sum_{\ell=1}^{L} r_\ell(h_1, h'_1).$$

Thus we need to compute $R_\infty(h_1, h'_1)$ to obtain $K(G, G')$.

Now let us restate this problem in terms of linear system theory (Rugh, 1995). The following recursive relationship holds between r_k and r_{k-1} ($k \geq 2$):

$$r_k(h_1, h'_1) = \sum_{i,j} t(i, j, h_1, h'_1) r_{k-1}(i, j). \quad (7)$$

Using (7), the recursive relationship for R_L also holds as follows:

$$\begin{aligned} R_L(h_1, h'_1) &= r_1(h_1, h'_1) + \sum_{k=2}^{T} r_k(h_1, h'_1) \\ &= r_1(h_1, h'_1) + \sum_{k=2}^{T} \sum_{i,j} t(i, j, h_1, h'_1) r_{k-1}(i, j) \\ &= r_1(h_1, h'_1) + \sum_{i,j} t(i, j, h_1, h'_1) R_{L-1}(i, j). \quad (8) \end{aligned}$$

Thus R_L is perceived as a discrete-time linear system (Rugh, 1995) evolving as the time L increases. Assuming that R_L converges, (see Sec. 3.4 for the convergence condition), we have the following equilibrium equation:

$$R_\infty(h_1, h'_1) = r_1(h_1, h'_1) + \sum_{i,j} t(i, j, h_1, h'_1) R_\infty(i, j) \quad (9)$$

Therefore, the computation of the marginalized kernel finally comes down to solving linear simultaneous equations (9) and substituting the solutions into (6).

Computing the kernel requires solving a linear equation with a $|G||G'| \times |G||G'|$ coefficient matrix. However, the matrix is actually sparse because the number of non-zero elements is less than $c^2|G||G'|$ where c is the maximum degree. Therefore, we can employ various kinds of efficient numerical algorithms that exploit sparsity (Barrett et al., 1994). In our implementation, we employed a simple iterative method that updates current solutions by using (8) until convergence starting from $R_1(h_1, h'_1) = r_1(h_1, h'_1)$.

3.4. Convergence Condition

The convergence condition needed to justify (9) is described as follows:

Theorem 1. *The infinite sequence* $\lim_{L\to\infty} R_L(h_1, h'_1)$ *converges for any* $h_1 \in \{1, \cdots, |G|\}$ *and* $h'_1 \in \{1, \cdots, |G'|\}$, *if the following inequality holds for* $i_0 \in \{1, \cdots, |G|\}$ *and* $j_0 \in \{1, \cdots, |G'|\}$,

$$\sum_{i=1}^{|G|} \sum_{j=1}^{|G'|} t(i, j, i_0, j_0) q(i, j) < q(i_0, j_0). \quad (10)$$

(proof) $R_\ell(h_1, h'_1)$ can also be described as

$$\begin{aligned} R_\ell(h_1, h'_1) &= q(h_1, h'_1) + \\ &\sum_{i=2}^{\ell} \sum_{h_2,h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3,h'_3} t(h_3, h'_3, h_2, h'_2) \times \left(\cdots \left(\sum_{h_i,h'_i} t(h_i, h'_i, h_{i-1}, h'_{i-1}) q(h_i, h'_i) \right) \cdots \right) \right) \\ &:= q(h_1, h'_1) + \sum_{i=2}^{\ell} u_i(h_1, h'_1), \end{aligned}$$

where u_i ($i \geq 2$) denotes the i-th term of the series. According to the ratio test, this series converges if $\limsup_{i\to\infty} |u_i(h_1, h'_1)|/|u_{i-1}(h_1, h'_1)| < 1$. Since $K(v, v')$ and $K(e, e')$ are nonnegative as previously defined, $u_i(h_1, h'_1) \geq 0$, and thus what we need to prove is $\limsup_{i\to\infty} u_i(h_1, h'_1)/u_{i-1}(h_1, h'_1) < 1$. For this purpose, it is sufficient to show

$$\frac{u_i(h_1, h'_1)}{u_{i-1}(h_1, h'_1)} < 1 \quad (11)$$

for all i. The two terms in (11) are written as

$$u_i(h_1, h'_1) = \sum_{h_{i-1},h'_{i-1}} a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) \times \sum_{h_i,h'_i} t(h_i, h'_i, h_{i-1}, h'_{i-1}) q(h_i, h'_i), \quad (12)$$

$$u_{i-1}(h_1, h'_1) = \sum_{h_{i-1},h'_{i-1}} a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) q(h_{i-1}, h'_{i-1}), \quad (13)$$

where

$$a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) \quad := \sum_{h_2,h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3,h'_3} t(h_3, h'_3, h_2, h'_2) \times \left(\cdots \left(\sum_{h_{i-2},h'_{i-2}} t(h_{i-2}, h'_{i-2}, h_{i-3}, h'_{i-3}) \times t(h_{i-1}, h'_{i-1}, h_{i-2}, h'_{i-2}) \right) \cdots \right) \right).$$

Notice that $a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) \geq 0$ as well. Substituting (12) and (13) into (11), we have

$$\sum_{h_{i-1},h'_{i-1}} a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) \times \sum_{h_i,h'_i} t(h_i, h'_i, h_{i-1}, h'_{i-1}) q(h_i, h'_i) < \sum_{h_{i-1},h'_{i-1}} a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1) q(h_{i-1}, h'_{i-1}).$$

Both sides of this inequality are the linear combinations of $a_{i-1}(h_{i-1}, h'_{i-1}, h_1, h'_1)$ with nonnegative coefficients. In order to prove the inequality, it is sufficient to prove the following inequalities for coefficients:

$$\sum_{i,j} t(i, j, i_0, j_0) q(i, j) < q(i_0, j_0), \quad \forall i_0, j_0,$$

which we have assumed to hold in (10). $\square$

The condition (10) seems rather complicated, but we can have a simpler condition, if the termination probabilities are constant over all vertices.

Lemma 1. *If $p_q(i) = p'_q(j) = \gamma$ for any i and j, the infinite sequence $\lim_{L\to\infty} R_L(h_1, h'_1)$ converges if*

$$K(v, v')K(e, e') < \frac{1}{(1-\gamma)^2}. \qquad (14)$$

(proof) Due to the assumption, $q(i, j) = q(i_0, j_0) = \gamma^2$. Thus it is sufficient to prove $\sum_{i,j} t(i, j, i_0, j_0) < 1$, that is,

$$\sum_{i,j} p_t(i|i_0) p'_t(j|j_0) K(v_i, v_j) K(e_{i_0 i}, e_{j_0 j}) < 1. \qquad (15)$$

Due to Relation (2), we observe that

$$\sum_{i,j} p_t(i|i_0) p'_t(j|j_0) = \sum_i p_t(i|i_0) \sum_j p'_t(j|j_0) = (1-\gamma)^2.$$

Thus (15) holds if all the coefficients satisfy $K(v_i, v_j) K(e_{i_0 i}, e_{j_0 j}) < 1/(1-\gamma)^2$. $\square$ Apparently, the above lemma holds for $\lambda > 0$ if $K(\cdot, \cdot) \leq 1$. Standard label kernels such as (3) and (4) satistfy this condition.

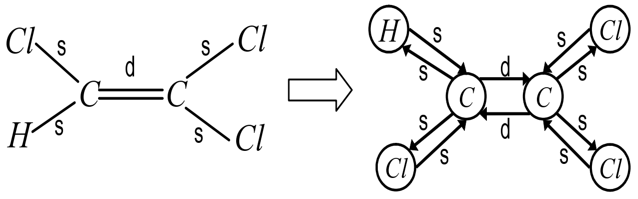

Figure 2. A chemical compound is conventionally represented as an undirected graph (left). Atom types and bond types correspond to vertex labels and edge labels, respectively. The edge labels 's' and 'd' denote single and double bonds, respectively. As our kernel assumes a directed graph, undirected edges are replaced by directed edges (right).

4. Experiments

We applied our kernel to prediction of the properties of chemical compounds. A chemical compound can naturally be represented as an undirected graph by considering the atom types as the vertex labels, e.g. C, Cl and H, and the bond types as the edge labels, e.g. *s* (single bond) and *d* (double bond). For our graph kernel, we replaced an undirected edge by two directed edges (Figure 2) since the kernel assumes directed graphs.

4.1. Pattern Discovery Algorithm

We compare our graph kernel with the pattern-discovery (PD) method by Kramer and De Raedt (2001) that is one of the best state-of-the-art methods in predictive toxicology. As in our graph kernel, each feature is constructed as the count that a particular label path appears in the graph[4]. There are other methods which count more complicated substructures such as subgraphs (Inokuchi et al., 2000), but we focus on Kramer and De Raedt (2001) whose features are similar to ours.

Assume that we have n graphs $G_1, \ldots, G_n$. Also let us define $\#(\boldsymbol{h}, G)$ as the number of appearances of a label path $\boldsymbol{h}$ in G. The PD method identifies a set of all label paths $\mathcal{H}$ which appear in more than m graphs:

$$\mathcal{H} = \{\boldsymbol{h} \mid \sum_{i=1}^{n} \delta\left(\#\left(\boldsymbol{h}, G_i\right) > 0\right) \geq m\},$$

where the parameter m is called the minimum support parameter. Furthermore, it is possible to add extra conditions, e.g., selecting only the paths frequent in a certain class, and scarce in the other classes. Then the feature vector of G based on the identified label paths

[4]Notice that the definition of label paths is different from ours in several points, e.g. a vertex will not be visited twice in a path. See Kramer and De Raedt (2001) for details.

is built as

$$G \to (\#(\boldsymbol{h}_1, G), \ldots, \#(\boldsymbol{h}_{|\mathcal{H}|}, G)), \tag{16}$$

whose dimensionality is the cardinality of $\mathcal{H}$. The PD method is useful for extracting comprehensive features. However, as the minimum support parameter gets smaller, the dimensionality of feature vectors becomes so huge that a prohibitive amount of computation is required. Therefore, the user has to control the minimum support parameter m, such that the feature space does not lose necessary information and, at the same time, computation stays feasible.

The PD method contrasts markedly with our method. Our kernel method puts emphasis on dealing with infinite, but less interpretable features, while the PD method trys to extract a relatively small number of meaningful features. Looking at the algorithms, our method is so simple that it is described by just one equation (9), while the PD method's algorithm is rather complicated (De Raedt & Kramer, 2001).

4.2. Datasets

We used two datasets, the PTC dataset (Helma et al., 2001) and the Mutag dataset (Srinivasan et al., 1996).

The PTC dataset is the results of the following pharmaceutical experiments. Each of 417 compounds is given to four types of test animals: Male Mouse (MM), Female Mouse (FM), Male Rat (MR) and Female Rat (FR). According to their carcinogenicity, each compound is assigned one of the following labels: $\{\mathrm{EE}, \mathrm{IS}, \mathrm{E}, \mathrm{CE}, \mathrm{SE}, \mathrm{P}, \mathrm{NE}, \mathrm{N}\}$ where CE, SE and P indicate "relatively active", and NE and N indicate "relatively inactive", and EE, IS and E indicate "can not be decided". In order to simplify the problem, we relabeled CE, SE and P as "positive", and NE and N as "negative". The task is to predict whether a given compound is positive or negative for each type of test animals. Thus we eventually had four two-class problems.

In the Mutag dataset, the task is defined to be a two-class classification problem to predict whether each of the 188 compounds has mutagenicity or not.

Each statistics of the datasets are summarized in Table 1.

4.3. Experimental Settings and Results

Assuming no prior knowledge, we defined the probability distributions for random walks as follows. The initial probabilities were simply uniform, i.e. $p_s(h) = 1/|G|, \forall h$. The termination probabilities were determined as a constant γ over all vertices. The transition probabilities $p_t(h|h_0)$ were set as uniform over adjacent vertices. We used Equation (3) as the label kernels. In solving the simultanous equations, we employed a simple iterative method (8). In our observation, 20-30 iterations were enough for convergence in all cases. For the classification algorithm, we used the voted kernel perceptron (Freund & Shapire, 1999), whose the performance is known to be comparable to SVMs. In the pattern discovery method, the minimum support parameter was determined as $0.5, 1, 3, 5, 10, 20\%$ of the number of compounds, and the simple dot product in the feature space (16) was used as a kernel. In our graph kernel, the termination probability γ was changed from 0.1 to 0.9.

Table 2 and Table 3 show the classification accuracies in the five two-class problems measured by leave-one-out cross validation. No general tendencies were found to conclude which method is better (the PD was better in MR, FR and Mutag, but our method was better in MM and FM). Thus it would be fair to say that the performances were comparable in this small set of experiments. Even though we could not show that our method is constantly better, this result is still appealing, because the advantage of our method lies in its simplicity both in concepts and in computational procedures.

Table 1. Several statistics of the datasets such as numbers of positive examples (#positive) and negative examples (#negative), maximum degree (max. degree), maximum size of graphs (max. $|G|$), average size of graphs (avg. $|G|$), and numbers of vertex labels ($|\Sigma|_v$) and edge labels ($|\Sigma|_e$).

	MM	FM	MR	FR	MUTAG
#POSITIVE	129	143	152	121	125
#NEGATIVE	207	206	192	230	63
MAX. $\lvert G\rvert$	109	109	109	109	40
AVG. $\lvert G\rvert$	25.0	25.2	25.6	26.1	31.4
MAX. DEGREE	4	4	4	4	4
$\lvert\Sigma\rvert_v$	21	19	19	20	8
$\lvert\Sigma\rvert_e$	4	4	4	4	4

Table 2. Classification accuracies (%) of the pattern discovery method. 'MinSup' shows the ratio of the minimum support parameter to the number of compounds m/n.

MINSUP	MM	FM	MR	FR	MUTAG
0.5%	60.1	57.6	61.3	**66.7**	88.3
1 %	**61.0**	**61.0**	**62.8**	63.2	87.8
3 %	58.3	55.9	60.2	63.2	**89.9**
5 %	60.7	55.6	57.3	63.0	86.2
10 %	58.9	58.7	57.8	60.1	84.6
20%	**61.0**	55.3	56.1	61.3	83.5

Table 3. Classification accuracies (%) of our graph kernel. The parameter γ is the termination probability of random walks, which controls the effect of the length of label paths.

γ	MM	FM	MR	FR	MUTAG
0.1	62.2	59.3	57.0	62.1	84.3
0.2	62.2	61.0	57.0	62.4	83.5
0.3	64.0	61.3	56.7	62.1	**85.1**
0.4	**64.3**	61.9	56.1	63.0	**85.1**
0.5	64.0	61.3	56.1	64.4	83.5
0.6	62.8	61.9	54.4	65.8	83.0
0.7	63.1	62.5	54.1	63.2	81.9
0.8	63.4	**63.4**	54.9	64.1	79.8
0.9	62.8	61.6	**58.4**	**66.1**	78.7

5. Conclusion

In this paper, we introduced a new kernel between graphs with vertex labels and edge labels in the framework of the marginalized kernel. We defined the kernel by using random walks on graphs, and reduced the computation of the kernel to solving a system of linear simulataneous equations. As contrasted with the pattern-discovery method, our kernel takes into account all possible label paths without computing feature values explicitly. The structure we dealt with in this paper is fairly general, as it includes sequences, trees, DAGs and so on. Thus, applications of this kernel are by no means limited to chemical compounds. Promising targets would be DNA and RNA sequences with remote correlations, HTML and XML documents, distance graphs of 3-D protein structures, and so on. Recently, we found that our kernel can also be interpreted as an instance of the rational kernels (Cortes et al., 2003) over probability semiring. In this interpretation, the probability distributions over a graph are considered as a weighted automaton. In furure works, we would like to explore how to deal with infinite dimensions when another semiring is adopted.

Appendix: Relation to Exponential and Geometric Kernels for Labeled Graphs

Here we review the exponential and geometric kernels between graphs proposed by Gärtner (2002) and describe the relation to our kernel. Although his kernels are defined in a quite different formulation from ours, we relate them by using the same notations.

His kernels are defined as the inner product of two $|\Sigma_v| \times |\Sigma_v|$ feature matrices $M(G)$ and $M(G')$,

$$K(G,G') = \sum_{a_1=1}^{|\Sigma_v|} \sum_{a_2=1}^{|\Sigma_v|} M_{a_1,a_2}(G) m'_{a_1,a_2}(G'),$$

where the feature matrix is defined as

$$M_{a_1,a_2}(G) = \sum_{\ell=1}^{\infty} \sum_{h_1,h_\ell} p(h_1,h_\ell)\delta(v_{h_1}=\sigma_{a_1})\delta(v_{h_\ell}=\sigma_{a_2}), \tag{17}$$

and $\Sigma_v = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma_v|}\}$.
Notice that the probability $p(h_1,h_\ell)$ is denoted as

$$p(h_1,h_\ell) = p_s(h_1)p_q(h_\ell) \times \sum_{h_2} \cdots \sum_{h_\ell} p_t(h_2|h_1)\cdots p_t(h_\ell|h_{\ell-1}).$$

Let T be a $|G| \times |G|$ matrix where $T_{ij} = p_t(j|i)$. Also, let S and Q be $|G| \times |\Sigma_v|$ matrices where $S_{ij} = p_s(i)\delta(v_i = \sigma_j)$ and $Q_{ij} = p_q(i)\delta(v_i = \sigma_j)$, respectively. By means of these matrices, the feature matrix is rewritten as

$$M(G) = \sum_{\ell=1}^{\infty} S^\top T^{\ell-1} Q.$$

In his settings, the parameters p_s, p_t and p_q do not have stochastic constraints, i.e. $p_s(i) = 1$ and $p_q(i) = 1$ for any vertex i. Therefore, $S = Q := L$, where $L_{ij} = \delta(v_i = j)$.

The feature vector of geometric kernel, $M(G) = \sum_{\ell=1}^{\infty} L^\top (\beta E)^\ell L$, is obtained by setting $T = \beta E$, where E is the adjacency matrix of G and β is a constant to assure convergence.
In the case of exponential kernel, M is defined as $M(G) = \sum_{\ell=1}^{\infty} (L^\top (\beta E)^{\ell-1} L)/(\ell-1)!$, which still results in a similar form.

As seen in (17), the primary difference from our approach is that they only take the vertex labels at both ends (h_1, h_ℓ) into account. Although this simplification allows to keep the size of the matrices small, it might be harmful when contiguous labels are essential as in chemical compounds. Another difference is that they do not use an infinite dimensional feature space. Although they also compute the infinite sum in Equation (17), the infinite sum is for obtaining $|\Sigma_v|^2$ dimensional feature vectors.

References

Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web: from relations to semistructured data and XML.* Morgan Kaufmann.

Bahlmann, C., Haasdonk, B., & Burkhardt, H. (2002). On-line handwriting recognition with support vector machines – a kernel approach. *Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)* (pp. 49–54).

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., & der Vorst, H. V. (1994). *Templates for the solution of linear systems: Building blocks for iterative methods, 2nd edition.* Philadelphia, PA: SIAM.

Collins, M., & Duffy, N. (2001). Convolution kernel for natural language. *Proc. of the 14th NIPS.*

Cortes, C., Haffner, P., & Mohri, M. (2003). Rational kernels. *Advances in Neural Information Processing Systems 15.* MIT Press. In press.

De Raedt, L., & Kramer, S. (2001). The levelwise version space algorithm and its application to molecular fragment finding. *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 853–862). Morgan Kaufmann.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Freund, Y., & Shapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning, 37,* 277–296.

Gärtner, T. (2002). Exponential and geometric kernels for graphs. *NIPS*02 Workshop on Unreal Data: Principles of Modeling Nonvectorial Data.* Available from http://mlg.anu.edu.au/unrealdata/.

Haussler, D. (1999). *Convolution kernels on discrete structures* (Technical Report UCSC-CRL-99-10). University of Calfornia in Santa Cruz.

Helma, C., King, R., Kramer, S., & Srinivasan, A. (2001). The Predictive Toxicology Challenge 2000-2001. *Bioinformatics, 17,* 107–108.

Inokuchi, A., Washio, T., & Motoda, H. (2000). An Apriori-based algorithm for mining frequent substructures from graph data. *Proc. of the 4th PKDD* (pp. 13–23).

Jaakkola, T., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology, 7,* 95–114.

Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2003). Learning semantic similarity. *Advances in Neural Information Processing Systems 15.* MIT Press. In press.

Kashima, H., & Inokuchi, A. (2002). Kernels for graph classification. *IEEE ICDM Workshop on Active Mining.* Maebashi, Japan.

Kashima, H., & Koyanagi, T. (2002). Kernels for semi-structured data. *Proc. of the 19th ICML* (pp. 291–298).

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. *Proc. of the 19th ICML* (pp. 315–322).

Kramer, S., & De Raedt, L. (2001). Feature construction with version spaces for biochemical application. *Proc. of the 18th ICML* (pp. 258–265).

Lafferty, J., & Lebanon, G. (2003). Information diffusion kernels. *Advances in Neural Information Processing Systems 15.* MIT Press. In press.

Leslie, C., Eskin, E., Weston, J., & Noble, W. (2003). Mismatch string kernels for svm protein classification. *Advances in Neural Information Processing Systems 15.* MIT Press. In press.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research, 2,* 419–444.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, Massachusetts: The MIT Press.

Rugh, W. (1995). *Linear system theory.* Prentice Hall.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels.* Cambridge, MA: MIT Press.

Shimodaira, H., Noma, K.-I., Nakai, M., & Sagayama, S. (2002). Dynamic time-alignment kernel in support vector machine. *Advances in Neural Information Processing Systems 14* (pp. 921–928). Cambridge, MA: MIT Press.

Srinivasan, A., Muggleton, S., King, R. D., & Sternberg, M. (1996). Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence, 85,* 277–299.

Takimoto, E., & Warmuth, M. K. (2002). Path kernels and multiplicative updates. *Proc. of the 15th Annual Conference on Computational Learning Theory* (pp. 74–89).

Tsuda, K., Kin, T., & Asai, K. (2002). Marginalized kernels for biological sequences. *Bioinformatics, 18,* S268–S275.

Vapnik, V. (1998). *Statistical learning theory.* New York: Wiley.

Vishwanathan, S., & Smola, A. (2003). Fast kernels for string and tree matching. *Advances in Neural Information Processing Systems 15.* MIT Press. In press.

Watkins, C. (2000). Dynamic alignment kernels. In A. Smola, P. Bartlett, B. Schölkopf and D. Schuurmans (Eds.), *Advances in large margin classifiers,* 39–50. MIT Press.