Extensions of Marginalized Graph Kernels

Pierre Mahé PIERRE.MAHE@ENSMP.FR
Ecole des Mines de Paris, 35 rue Saint Honoré, 77300 Fontainebleau, France

Nobuhisa Ueda UEDA@KUICR.KYOTO-U.AC.JP
Tatsuya Akutsu TAKUTSU@KUICR.KYOTO-U.AC.JP
Jean-Luc Perret LUC@KUICR.KYOTO-U.AC.JP
Bioinformatics Center, Kyoto University, Uji, Kyoto 611-0011, Japan

Jean-Philippe Vert JEAN-PHILIPPE.VERT@ENSMP.FR
Ecole des Mines de Paris, 35 rue Saint Honoré, 77300 Fontainebleau, France

Abstract

Positive definite kernels between labeled graphs have recently been proposed. They enable the application of kernel methods, such as support vector machines, to the analysis and classification of graphs, for example, chemical compounds. These graph kernels are obtained by marginalizing a kernel between paths with respect to a random walk model on the graph vertices along the edges. We propose two extensions of these graph kernels, with the double goal to reduce their computation time and increase their relevance as measure of similarity between graphs. First, we propose to modify the label of each vertex by automatically adding information about its environment with the use of the Morgan algorithm. Second, we suggest a modification of the random walk model to prevent the walk from coming back to a vertex that was just visited. These extensions are then tested on benchmark experiments of chemical compounds classification, with promising results.

1. Introduction

Many real-world data, such as natural language texts or molecular structures, can be represented as graphs. A number of applications, such as mining molecular databanks to predict activity or toxicology of potential drugs, require the analysis, comparison, and classification of these graphs. Among the many different ways to tackle this problem, two mainstream research directions have emerged during the last decades. One direction involves the use of graph algorithms to compare or classify graphs, for instance by finding maximal or frequent common subgraphs. Such approaches usually suffer from their computational complexity (NP-hardness of subgraph isomorphism, exponential number of potential subgraphs), and are usually based on heuristic or restricted to small-size graphs and databanks. The second mainstream direction, particularly in chemoinformatics, consists in transforming the graphs into vectors using molecular descriptors—what one would usually call *features* in the machine learning community—, before applying the whole panoply of statistical or machine learning tools to the vector representations. This usually requires the selection of a small number of features of interest.

An alternative direction has been explored recently by Kashima et al. (2003) and Gärtner et al. (2003), using the theory of positive definite kernels and kernel methods (Schölkopf & Smola, 2002). The authors introduce positive definite kernels between labeled graphs, based on the detection of common paths between different graphs. These kernels correspond to a dot product between the graphs mapped to an infinite-dimensional feature space, but can be computed in polynomial time with respect to the graph sizes. Encouraging experimental results suggest that this approach might be a valid alternative to the two mainstream directions.

These graph kernels, however, are subject to several limitations which we try to address in this contribution. First, the choice of representing implicitly each graph by the set of path probabilities under a

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

simple random walk model might be questioned. In many applications, such as chemoinformatics, subgraphs are believed to be more relevant features than paths. Moreover, the random walk model used is subject to "tottering", in the sense that it can move to one direction and instantly come back to its original position, resulting in redundant paths which might decrease the characterization of a given graph once mapped to the feature space of these graph kernels. Second, the graph kernel has a computational complexity roughly proportional to the product of the sizes of the two graphs to be compared, which results in slow implementation for real-world problem.

We propose two extensions of the original graph kernel, which try to address these issues. The first extension is to relabel each vertex automatically in order to insert information about the environment of each vertex in its label. This has both an effect in terms of feature relevance, because label paths contain information about their environment as well, and computation time, because the number of identical labeled paths significantly decreases. Second, we show how to modify the random walk model in order to remove totters, without increasing the complexity of the implementation. Each method is validated on a benchmark experiment of chemical compounds classification.

2. Marginalized graph kernel

In this section we define the basic notations and briefly review the graph kernel introduced in Kashima et al. (2003) and Gärtner et al. (2003).

2.1. Labeled graphs

A *labeled graph* $G = (V, E)$ is defined by a finite set of *vertices* V (of size $|V|$), a set of *edges* $E \subset V \times V$, and a labeling function $l : V \cup E \mapsto A$ which assigns a *label* $l(x)$ to any vertex or edge x. We assume below that a set of labels A has been fixed, and consider different labeled graphs. For a given graph $G = (V, E)$, we denote by $d(v)$ the number of edges emanating from the vertex v (i.e., the number of edges of the form (v, u)), and by $V^* = \bigcup_{n=1}^{\infty} V^n$ the set of finite-length sequences of vertices. A *path* $h \in V^*$ is a finite-length sequence of vertices $h = v_1 \dots v_n$ with the property that $(v_i, v_{i+1}) \in E$ for $i = 1, \dots, n-1$. We note $|h|$ the length of the path h, and $H(G) \subset V^*$ the set of all paths of a graph G. The labeling function $l : V \cup E \mapsto A$ can be extended as a function $l : H(G) \mapsto A^*$ where the label $l(h)$ of a path $h = v_1 \dots v_n \in H(G)$ is the succession of labels of the vertices and edges of the path: $l(h) = (l(v_1), l(v_1, v_2), l(v_2), \dots, l(v_{n-1}, v_n), l(v_n)) \in A^{2n-1}$.

2.2. Marginalized graph kernels

A positive definite kernel on a space $\mathcal{X}$ is a symmetric function $K : \mathcal{X}^2 \Rightarrow \mathbb{R}$ that satisfies $\sum_{i,j=1}^{n} a_i a_j K(x_i, x_j)$ for any choice of n points $x_1, \dots, x_n \in \mathcal{X}$ and coefficients $a_1, \dots, a_n \in \mathbb{R}$. Defining a kernel on a space $\mathcal{X}$ paves the way to the use of kernel methods (Schölkopf & Smola, 2002) for classification, regression of clustering, for example.

In order to define a kernel function $K(G_1, G_2)$ between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, Kashima et al. (2003) proposed to use the following formula:

$$K(G_1, G_2) = \sum_{(h_1,h_2) \in V_1^* \times V_2^*} p_1(h_1) p_2(h_2) K_L\left(l(h_1), l(h_2)\right), \quad (1)$$

where p_1 and p_2 are probability distributions on V_1^* and V_2^* and the function $K_L : A^* \times A^* \mapsto \mathbb{R}$ in (1) is a kernel between label sequences. Equation (1) can be seen as a marginalized kernel (Tsuda et al., 2002) and is therefore a valid kernel on the set of graphs.

Kashima et al. (2003) focus on the particular case where, for a graph $G = (V, E)$, the kernel between label sequences in (1) is the Dirac kernel:

$$K_L(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2, \\ 0 & \text{otherwise,} \end{cases}$$

and where the probability p on V^* used in the kernel definition (1) factorizes as:

$$p(v_1 \dots v_n) = p_s(v_1) \prod_{i=2}^{n} p_t(v_i | v_{i-1}). \quad (2)$$

In order to ensure that (2) defines a probability distribution on V^* (i.e., $\sum_{v \in V^*} p(v) = 1$), we must impose constraints on p_s and p_t. This can be done, for example, by choosing parameters $0 < p_q(v) < 1$ for $v \in V$, an initial probability distribution p_0 on V ($\sum_{v \in V} p_0(v) = 1$), a transition matrix p_a on $V \times V$ ($\sum_{u \in V} p_a(u|v) = 1$ for $v \in V$) positive only along edges ($p_a(v|u) > 0 \Rightarrow (u, v) \in E$), and by setting, for any $u, v \in V^2$,

$$\begin{cases} p_s(v) = p_0(v) p_q(v), \\ p_t(u|v) = \frac{1 - p_q(v)}{p_q(v)} p_a(u|v) p_q(u). \end{cases}$$

Under these conditions it can easily be checked that (2) is a probability distribution on V^* corresponding to a random walk on the graph with initial distribution p_0, transition probability p_a, and stopping probability p_q at each step. In particular, this implies that only paths have positive probabilities under $p : p(h) > 0 \Rightarrow h \in H(G)$.

2.3. Graph kernel computation

While the kernel definition (1) involves a summation over an infinite number of paths, it can be computed efficiently using product graphs and matrix inversions (Gärtner et al., 2003), as briefly recalled below. Given two labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *product graph* is defined as the labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose vertices $\mathcal{V} \subset V_1 \times V_2$ are pairs of vertices with identical labels ($(v_1, v_2) \in \mathcal{V}$ iff $l(v_1) = l(v_2)$), and an edge connects the vertices (u_1, u_2) and (v_1, v_2) iff $(u_i, v_i) \in E_i$, for $i = 1, 2$, and $l(u_1, v_1) = l(u_2, v_2)$. Let us now defines a functional π on the set of paths $H(\mathcal{G})$ by

$$\pi\left((u_1, v_1)(u_2, v_2)\dots(u_n, v_n)\right) = \pi_s\left(u_1, v_1\right) \prod_{i=2}^{n} \pi_t\left((u_i, v_i)|(u_{i-1}, v_{i-1})\right),$$

with

$$\begin{cases} \pi_s(u_1, u_2) = p_s^{(1)}(u_1) p_s^{(2)}(u_2), \\ \pi_t((v_1, v_2)|(u_1, u_2)) = p_t^{(1)}(v_1|u_1) p_t^{(2)}(v_2|u_2), \end{cases}$$

where $p_s^{(1)}$ and $p_t^{(1)}$ (resp. $p_s^{(2)}$ and $p_t^{(2)}$) are the functions used to define the probabilities of random paths in (2) on the graph G_1 (resp. G_2). It can then be shown that:

$$K(G_1, G_2) = \sum_{h \in H(\mathcal{G})} \pi(h).$$

If one now defines the $|\mathcal{V}|$-dimensional vector $\pi_s = (\pi_s(v))_{v \in \mathcal{V}}$ and the $|\mathcal{V}| \times |\mathcal{V}|$ transition matrix $\Pi_t = (\pi_t(v|u))_{(u,v) \in \mathcal{V}^2}$, then it can be checked that:

$$\sum_{h \in H(\mathcal{G}), |h|=n} \pi(h) = \pi_s^\top \Pi_t^n \mathbf{1},$$

where $\mathbf{1}$ is the $|\mathcal{V}|$-dimensional vector with all entries equal to 1, and therefore:

$$K(G_1, G_2) = \sum_{n=1}^{\infty} \left(\sum_{h \in H(\mathcal{G}), |h|=n} \pi(h) \right) = \pi_s^\top \left(I - \Pi_t\right)^{-1} \mathbf{1}.$$

While the size of the matrix Π_t can be in the worst case $|V_1||V_2|$, it is sparse and its inversion can be approximated by a finite sum of the first terms in the power series expansion of the matrix inverse, involving only sparse matrix products and resulting in a complexity of order $O(|\Pi_t| d^N)$, where $|\Pi_t|$ is the number of non-zero elements in the matrix Π_t, d is the maximum degree of the product graph, and N is the number of iterations used to approximate the inverse (Kashima et al., 2003; Smola & Kondor, 2003).

3. Label enrichment with the Morgan index

We observed in practice that the computation of the graph kernel is time-consuming (about one day on a recent desktop PC to compute a Gram matrix for a dataset of 400 molecules), and largely dominates the cost of training the SVM (which takes a few seconds on the same dataset). Moreover the computation time increases drastically when more sophisticated random walk models are used (see Section 4). This suggests that speeding up the kernel computation is required for real-world applications where datasets of several 10,000's molecules are common. Second, one might expect the search of common paths to be too naive to detect interesting patterns between chemical compounds.

In order to overcome both issues simultaneously, we propose to increase the specificity of labels, by including contextual information about the vertices in their labels. This has two important consequences. First, as the label specificity increases, the number of common label paths between graphs automatically decreases, which shortens the computation time. Second, this is likely to increase the relevance of the features used to compare graphs, as paths are replaced by paths labeled with their environment.

A simple and fast label enrichment procedure is given by the following iterative process. Each vertex is initially labeled with the integer 1. Then, at each iteration, the label of a vertex is the sum of its direct neighbor's labels. Mathematically, if M_n denotes the vector of labels, this reads $M_0 = \mathbf{1}$ and $M_{n+1} = AM_n$, where A is the graph adjacency matrix. Figure 1 illustrates the first two iterations of this process, which after n iterations computes the number of different paths of length n starting from each node.

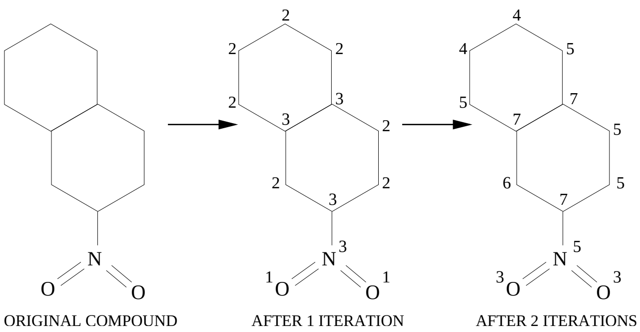

Figure 1. Morgan index process

While different label enrichment procedures can be defined, we focus below on this particular procedure that results in a family of kernels $(K_n)_{n\geq 0}$, indexed by the number of iterations performed in the relabeling process. Indeed, this particular scheme is well-known in chemoinformatics under the name of *Morgan index* (Morgan, 1965), and is considered a good and fast solution to detect graph isomorphism and determine canonical representations of molecules.

4. Preventing totters

A second avenue to modify the original graph kernel is to modify the probability (2). This probability is the distribution of a 1st-order Markov random walk along the edges of the graph, killed with some probability after each step. We propose to modify the random walk model to prevent "totters", that is, to avoid any path of the form $h = v_1, \ldots, v_n$ with $v_i = v_{i+2}$ for some i. The motivation here is that such excursions are likely to add noise to the representation of the graph. For example, the existence of a path with labels `C-C-C` might either indicate the presence of a succession of 3 `C`-labeled vertices in the graph, or just a succession of 2 `C`-labeled vertices visited by a tottering random walk. By preventing totters, the second possibility disappears.

4.1. Modification of the random walk

A natural way to carry out this modification is to keep the general kernel definition (1) but modify the probability model (2) as follows:

$$p(v_1 \ldots v_n) = p_s(v_1)p_t(v_2|v_1)\prod_{i=3}^{n} p_t(v_i|v_{i-2}, v_{i-1}), \quad (3)$$

where $p_i(.)$, $p_t(.|.)$, and $p_t(.|.,.)$ satisfy for any $(u, v) \in V^2$:

$$\begin{cases} p_s(v) = p_0(v)p_q^{(0)}(v), \\ p_t(u|v) = \frac{1-p_q^{(0)}(v)}{p_q^{(0)}(v)}p_a(u|v)p_q(u), \\ p_t(u|w, v) = \frac{1-p_q(v)}{p_q(v)}p_a(u|w, v)p_q(u). \end{cases}$$

Here we assume that $0 < p_q(v), p_q^{(0)}(v) \leq 1$ for each vertex v, $p_a(\cdot|v)$ is a probability on V that is only positive on the neighbors of v, and $p_a(\cdot|w, v)$ is a probability on V that is only positive on the neighbors of v different from w. This model is simply the distribution of a 2nd-order Markov random walk, killed at each step with some probability $p_q(v)$ (or $p_q^{(0)}(v)$ after the first vertex, see section 5.1), which can not follow excursions of the form $u \to v \to u$. In other words, only paths belonging to

$$H_0(G) = \{h = v_1 \ldots v_n : v_i \neq v_{i+2}, i = 1, \ldots, n-2\}, \quad (4)$$

can have a positive probability under this model. Given this new random walk model, the function (1) is still a valid kernel, but the implementation described in section 2.3 can not be used directly anymore.

4.2. Computation of the new kernel

We now derive an explicit way to perform the computation of the kernel (1) under the model (3), by transforming the initial graphs into new graphs where the 2nd-order random walk (3) factorizes as a first-order Markov process (2).

More precisely, for a graph $G = (V, E)$, let the transformed graph $G' = (V', E')$ be defined by

$$V' = V \cup E,$$

and

$$\begin{aligned} E' = &\{(v, (v, t)) \,|v \in V, (v, t) \in E\} \\ &\cup \{((u, v), (v, t)) \,|\, (u, v), (v, t) \in E, u \neq t\}. \end{aligned} \quad (5)$$

The transformed graph G' is labeled as follows. For a node $v' \in V'$ the label is $l'(v') = l(v')$ if $v' \in V$, or $l'(v') = l(v)$ if $v' = (u, v) \in E$. For an edge $e' = (v'_1, v'_2)$ between two vertices $v'_1 \in V \cup E$ and $v'_2 \in E$, the label is simply given by $l'(e') = l(v'_2)$. The construction of G' and its labeling are illustrated in figure 2.

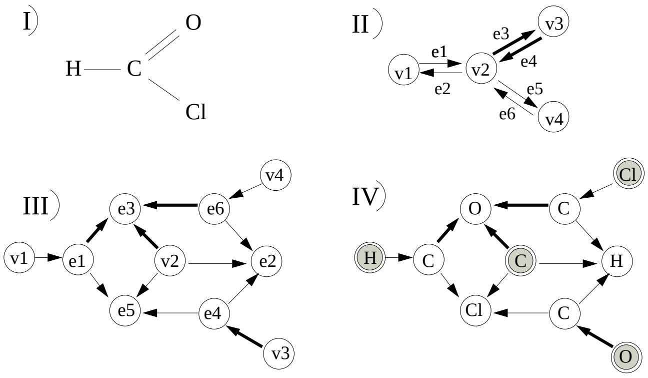

Figure 2. The graph transformation. I) The original molecule. II) The corresponding graph $G = (V, E)$. III) The transformed graph. IV) The labels on the transformed graph. Note that different widths stand for different edges labels, and gray nodes are the nodes belonging to V.

The vertices of the transformed graph G' can be either edges or vertices of the original graph G. Among all paths $H(G')$ on G', let us consider the subset of paths that start on a vertex, that is the set

$$H_1(G') = \{h' = v'_1, \ldots, v'_n \in H(G') : v'_1 \in V\}. \quad (6)$$

Note that from the definition of the product graph edges, it is easy to check that any path $h' = v'_1 \dots v'_n \in H(v')$ starting with a vertex $v_1 \in V$ must be made of edges: $v_i \in E, i = 2, \dots, n$. This construction is illustrated in figure 2.

Given (3), let the functional $p' : (V')^* \to \mathbb{R}$ be derived from (3) by:

$$p'(v'_1 \dots v'_n) = p'_s(v'_1) \prod_{i=2}^{n} p'_t(v'_i | v'_{i-1}), \tag{7}$$

with

$$p'_s(v') = \begin{cases} p_s(v') & \text{if } v' \in V, \\ 0 & \text{if } v' \in E, \end{cases}$$

and

$$p'_t(v'|u') = \begin{cases} p_t(v|u') & \text{if } u' \in V \text{ and } v' = (u', v) \in E, \\ p_t(v|t,u) & \text{if } u' = (t,u) \in E \\ & \text{and } v' = (u,v) \in E. \end{cases}$$

Finally, let us consider the map $f : H_0(G) \to (V')^*$ defined by:

$$f(v_1 \dots v_n) = v'_1 \dots v'_n,$$

with

$$\begin{cases} v'_1 = v_1 \in V, \\ v'_i = (v_{i-1}, v_i) \in E, \text{ for } i = 2, \dots, n. \end{cases} \tag{8}$$

Then the following result, whose proof is postponed to Appendix 7, holds:

Theorem 1 *f is a bijection between $H_0(G)$ and $H_1(G')$, and for any path $h \in H_0(G)$ we have*

$$\begin{cases} l(h) = l'(f(h)), \\ p(h) = p'(f(h)). \end{cases}$$

We can immediately deduce the following

Corollary 1 *For any two graphs G_1 and G_2, the kernel (1) can be expressed in terms of the transformed graphs G'_1 and G'_2 by:*

$$K(G_1, G_2) = \sum_{(h'_1, h'_2) \in (V'_1)^* \times (V'_2)^*} p'_1(h'_1) p'_2(h'_2) K_L(l'(h'_1), l'(h'_2)). \tag{9}$$

This shows that computing the $K(G_1, G_2)$ under the 2nd-order Markov model (3) for the random walk is equivalent to computing a kernel between the transformed graphs G'_1 and G'_2 under a 1st-order Markov random walk (7) which can be carried out as described in section 2.3 with an increased complexity.
More precisely, if we let $G' = (V', E')$ be the graph resulting of the transformation of a graph $G = (V, E)$, then $|V'| = |V| + |E|$. Hence, the graph product between two transformed graphs G'_1 and G'_2 may at worst be of size $(|V_1| + |E_1|) \times (|V_2| + |E_2|)$, which raises the complexity of the kernel from $O(|V_1||V_2|)$ to $O((|V_1| + |E_1|)(|V_2| + |E_2|))$.

5. Experiments

We tested both graph kernel extensions on two benchmark experiments of chemical compound classification already used in Kashima et al. (2003). We parametrize the 1st-order Markov random walk model by a single parameter $p_q < 1$ as follows. For any vertex v, we set $p_s(v) = 1/|V|$ and $p_q(v) = p_q < 1$. For any pair of vertices, we set $p_a(v|u) = 1/d(u)$ is $(u, v) \in E$, 0 otherwise.

Similarly the 2nd-order Markov model (3) is defined as follows. For any vertex v, $p_s(v) = 1/|V|$, $p_q^{(0)}(v) = p_q$, and $p_q^(v) = 1$ if $d(v) = 1$, 0 otherwise. For any pair (u, v), $p_a(v|u) = 1/d(u)$ is $(u, v) \in E$, 0 otherwise. Finally, for any vertices $(u, v, w))$, we set $p_a(u|w, v) = 1/(d(u) - 1)$ if $(v, u) \in E$, 0 otherwise.

The main differences between the 1st- and 2nd-order Markov models concern the functional $p_q(v)$, $p_q^{(0)}(v)$, and $p_a(u|w, v)$. Indeed we have to explicitly kill random walks when reaching a node with only one neighbor, except for the first step, because in this case, the only possibility to continue the walk is to "totter" to the previous node. The definition of $p_a(u|w, v)$ also reflects the modification required to prevent totters: the number of possible edges to follow from a node v is only $d(v) - 1$, because one edge has already been used to reach v.

The experiments described below are classification experiments, that we carried out with a support vector machine based on the different kernels tested. Each kernel was implemented in C++, and we used the free and publicly available GIST[1] implementation of SVMs.

5.1. MUTAG dataset

We tested the graph kernel extensions on the MUTAG dataset (Debnath et al., 1991), which contains 230 chemical compounds (aromatic and heteroaromatic ni-

[1] `http://microarray.cpmc.columbia.edu/gist`

tro compounds) tested for mutagenicity on *Salmonella typhimurium*. We focused on the possibility to infer mutagenicity on a subset of 188 compounds considered to be amenable to classification by Debnath et al. (1991), split into 125 positive examples (positive levels of log mutagenicity) and 63 negative examples. Each chemical compound is represented as a graph with atoms as vertices and covalent bonds as edges. We evaluate each kernel by the leave-one-out error.

1st- and 2nd-order model comparison: We first compared the classification accuracy of the graph kernels corresponding to the 1st- and 2nd-order Markov random walks, for different values of p_q. Results are shown in Table 1, where we observe that the change from a 1st- to a 2nd-order Markov model has no significant effect on the success rates (89.9% to 90.4% accuracy). The best perfomance is however reached at a higher value of p_q for the 2nd-order Markov model, which suggests that shorter path information is used in this case.

Augmented vertex labels: Table 2 shows classification results using the original random walk model and different iterations of the Morgan index. Interestingly, success rates increase for the first 2 or 3 iterations of the Morgan algorithm and decrease afterward. We can also note that after the 10th iteration, success rates stay steady. A careful inspection of the kernel values show that after the 10th iteration or so, only similar structures have non-zero kernel values in this dataset. This does not mean that the kernel is diagonal, because molecules might have the same structure with different atoms at some positions. On the other hand, Figure 3 shows the computation time to compute the kernel Gram matrices with various Morgan indices. As expected, a drastic decrease is observed at each iteration of the Morgan index. For example, the time is reduced by a factor of 240 after two iterations.

New random walk model with augmented vertex labels: Finally, Table 3 shows classification results obtained from the combination of the 2nd-order random walk model together with Morgan index description of molecules. We note that the introduction of the Morgan index using the second order Markov model enables to increase classification accuracy, but results do not get higher than those presented in Table 2.

5.2. PTC data set

The Predictive Toxicology Challenge (Toivonen et al., 2003) data is a dataset of molecules dealing with

Probability p_q	0.1	0.4	0.7
1st-order model	89.9	88.8	87.2
2nd-order model	88.8	90.4	87.2

Table 1. Classification results (in percents) for the MUTAG dataset using different random walk models

Probability p_q	0.1	0.4	0.7
Original accuracy	89.9	88.8	87.2
1st iteration	**91**	89.4	89.4
2nd iteration	**91**	89.9	**90.4**
3rd iteration	88.3	**91**	**90.4**
4th iteration	86.7	87.2	87.2
5th iteration	86.2	87.8	88.3
6th iteration	81.4	85.6	86.2
7th iteration	76.6	81.4	81.4
8th iteration	75.5	77.1	79.8
9th iteration	75	76.1	75.5
10th iteration	74.5	75	75.5
11 - 20th iteration	74.5	74.5	74.5

Table 2. Classification results (in percents) for the MUTAG dataset introducing different Morgan indices in the labeling of atoms.

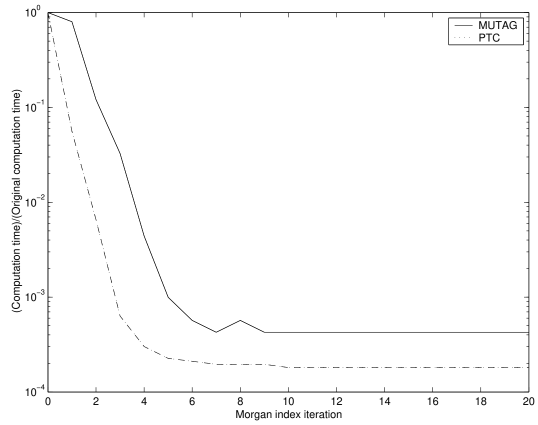

Figure 3. Computation times using different iterations of the Morgan Index process for MUTAG and PTC datasets (the y-axis is on a log-scale).

Probability p_q	0.1	0.4	0.7
1st iteration	88.8	90.4	89.4
2nd iteration	89.9	89.9	90.4

Table 3. Classification results (in percents) for the MUTAG dataset using second order random walk Markov model together with Morgan index description of molecules.

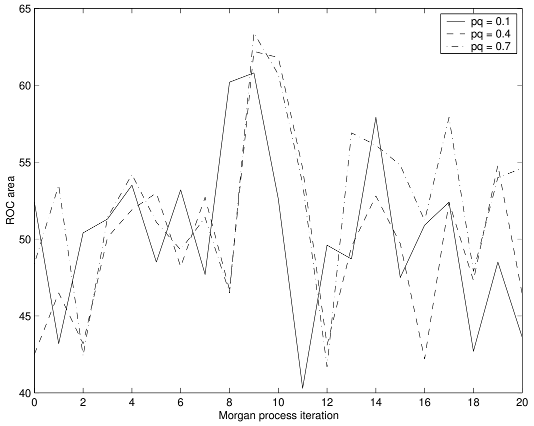

Figure 4. ROC area evolution with introduction of the Morgan index for different values of p_q in the PTC dataset, Female Rats

carcinogenicity. It is made of a training set (417 molecules) and a test set (184 molecules) gathering results of carcinogenicity tests clinically performed on different rodents (namely male mice, female mice, male rats and female rats). A standard training/test procedure thus enables to build four predictive models.

By lack of space we just show the effect of using the Morgan index on time and classification accuracy. As for the MUTAG dataset, the computation time of the kernel significantly decreases with each iteration of the Morgan index (figure 3). Performance, measured by the normalized area under the ROC curve plotting the number true positives as a function of false positives (Gribskov & Robinson, 1996), is shown in figure 4 for different Morgan index iterations and different values of p_q. This dataset is known to be difficult, and values around 50 indicate that the performance is not better than a random classifier. We observe that the best performances are obtained after $8-10$ iterations of the Morgan index computation. This contrasts with the results obtained with the MUTAG dataset (where the best results are obtained after 2-3 iterations). A careful analysis of the kernel matrices show that on the PTC dataset, the kernel entries stop varying after about 12 iterations. At 9 iterations, about 3% of the kernel Gram matrix (apart from the diagonal) are non-zero, corresponding to strong graph similarity. This suggests that on this particular dataset, known to be difficult to classify, the best results are obtained when only close similarity between graph is used in the inference process.

6. Discussion

We proposed two modifications of the graph kernel by Kashima et al. (2003). **The vertex label enrichment** could only slightly improve classification accuracy, while drastically reducing the computing time. This performance in calculation speed opens the way to use kernel methods on large databases of graphs like chemical databases. For drug design applications for instance, the databases to be analyzed usually consist of several hundred thousand of molecules, and computation aspects are of crucial importance.
On the other hand, these experiments reveal that an optimal iteration of the process is to be found in relation with the data considered. When this optimal iteration is not met, patterns are either not specific enough (for few iterations), or too specific (for many iterations) to best detect the similarity between graphs. The fact that this number is different for MUTAG and PTC (the 2nd or 3rd iteration for MUTAG, and 8-10th for PTC) is probably due to the different nature of these datasets : MUTAG is a considered a "simple" dataset, where features such as short paths are known to be relevant, while PTC is a "difficult" dataset (Toivonen et al., 2003), where toxicity can only be inferred locally between very similar molecules.

Removing totters from random walks did not improve classification performance uniformly. At small end probability (p_q) the classification performance was reduced compared to the original random walk model. At higher p_q however, the new random walk model increased performance compared to the original random walk model. At high p_q, paths are shorter and totters may represent a relatively high proportion of the random walks. The elimination of totters in these short paths may explain the increased classification performance. At lower p_q, on the other hand, random walks are longer on average, so that totters might not represent a large proportion of the random walks. In these longer paths, removing totters might underrepresent important local features in molecules and lead to the reduced classification performance observed. The new random walk model, while it did not generally increase classification performance for the MUTAG dataset, might prove very useful in classification problems where small subgraphs are important for graph classification.

As a conclusion, it is worth observing that the results presented in this paper compare favorably to several state-of-the-art algorithms presented in (King et al., 1996) based on a vectorization of molecules via molecular descriptors, as summarized in table 4.

Future work will focus on studying different label en-

Lin. Reg.	Neural Nets	Decision Trees	I.L.P.
89.3%	89.4%	88.3%	87.8%

Table 4. Classification results (in percents) for the MUTAG dataset using different algorithms. Results taken from (King et al., 1996)

richment schemes, either by different classical graph indices or by introducing chemical knowledge in the general kernel formulation in order to define a specific molecular kernel. We can for instance take into account the different types of atoms to define a more powerful label kernel K_L in equation (1), and replace the uniform random walk model so that it stresses the influence of rare, or a priori relevant atoms.

7. Appendix: Proof of Theorem 1

For any path $h = v_1 \dots v_n \in H_0(G)$, let $f(h) = v'_1 \dots v'_n$ defined by (8). By definition (5), $(v'_1, v'_2) = (v_1, (v_1, v_2)) \in E'$, and $\left(v'_i, v'_{i+1}\right) = ((v_{i-1}, v_i), (v_i, v_{i+1})) \in E'$ because $v_{i+1} \neq v_{i-1}$ for $i > 1$. Hence $f(h)$ is a path in G'. Moreover $v'_1 \in V$ and $v'_i \in E$ by (8), hence $f(h) \in H_1(G')$ by (6).

Conversely, for any $h' = v'_1 \dots v'_n \in H_1(G')$, we have $v'_1 = v_1 \in V$ and by easy induction using the definition of edges (5), $v'_i = (v_{i-1}, v_i) \in E$ with $v_{i-1} \neq v_{i+1}$. Hence $h' = f(h)$ with $h = v_1 \dots v_n \in H_0(G)$, therefore f is surjective. By definition of f (8), it is also clear that $f(h) = f(h') \Rightarrow h = h'$. f is therefore a bijection from $H_0(G)$ onto $H_1(G')$.

By definition of the labeling l' on G', we obtain for any $h = v_1, \dots, v_n \in H_0(G)$:

$$\begin{aligned} l'(f(h)) &= l'\left(v_1, (v_1, v_2), \dots, (v_{n-1}, v_n)\right) \\ &= l(v_1)l(v_2)\dots l(v_n) = l(h). \end{aligned}$$

We also obtain, from the definition of p':

$$\begin{aligned} p'(f(h)) &= p'\left(v_1, (v_1, v_2), \dots, (v_{n-1}, v_n)\right) \\ &= p'_s(v_1)p'_t\left((v_1, v_2)|v_1\right) \\ &\qquad \prod_{i=3}^{n} p'_t\left((v_{i-1}, v_i)|(v_{i-2}, v_{i-1})\right) \\ &= p_s(v_1)p_t(v_2|v_1)\prod_{i=3}^{n} p_t\left(v_i|v_{i-2}, v_{i-1}\right) \\ &= p(h) \quad \square \end{aligned}$$

Acknowledgments

We thanks anonymous reviewers for several useful comments. This work was supported by a French-Japanese *SAKURA* grant. JLP is partly supported by a grant from the *Swiss National Science Foundation*.

References

Debnath, A. K., de Compadre, R. L. L., Debnath, G., Schusterman, A., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 786–797.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. *Proc. of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*. Heidelberg: Springer-Verlag.

Gribskov, M., & Robinson, N. L. (1996). Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, *20*, 25–33.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 321–328). AAAI Press.

King, R. D., Muggleton, S. H., Srinivasan, A., & Sternberg, M. J. E. (1996). Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proc. Natl. Acad. Sci. USA*, *93*, 438–442.

Morgan, H. (1965). The generation of unique machine description for chemical structures - a technique developped at chemical abstracts service. *Journal of Chemical Documentation*, 107–113.

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

Smola, A., & Kondor, I. (2003). Kernels and regularization on graphs. *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Colt/Kernel 2003* (pp. 144–158). Springer.

Toivonen, H., Srinivasan, A., King, R. D., Kramer, S., & Helma, C. (2003). Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics*, *19*, 1183–1193.

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