Virtual Screening with Support Vector Machines and Structure Kernels

Pierre Mahé[1] and Jean-Philippe Vert[*,2]

[1]*Xerox Research Center Europe, 6, Chemin de Maupertuis, 38240 Meylan, France*

[2]*Centre for Computational Biology, Ecole des Mines de Paris, ParisTech, 35 rue Saint Honoré, 77305, Ontainebleau, France*

[3]*Institut Curie, 75248 Paris, France*

[4]*INSERM U900, 75248 Paris, France*

Abstract: Support vector machines and kernel methods have recently gained considerable attention in chemoinformatics. They offer generally good performance for problems of supervised classification or regression, and provide a flexible and computationally efficient framework to include relevant information and prior knowledge about the data and problems to be handled. In particular, with kernel methods molecules do not need to be represented and stored explicitly as vectors or fingerprints, but only to be *compared* to each other through a comparison function technically called a *kernel*. While classical kernels can be used to compare vector or fingerprint representations of molecules, completely new kernels were developed in the recent years to directly compare the 2D or 3D structures of molecules, without the need for an explicit vectorization step through the extraction of molecular descriptors. While still in their infancy, these approaches have already demonstrated their relevance on several toxicity prediction and structure-activity relationship problems.

Keywords: Virtual screening, graph kernels, support vector machines.

INTRODUCTION

Computational approaches play an increasingly important role in modern drug discovery. In particular, accurate predictive models accounting for the biological activity and drug-likeliness of candidate molecules can help in the identification of promising molecules and screening for various side-effects, leading to substantial savings in terms of time and costs for the development of new drugs. Such predictive models aim at inferring a relationship between the structure of a molecule and its biological and chemical properties, including toxicity, pharmacokinetics and activity against a biological target. The development of high-throughput technologies to assay such properties for large numbers of candidate molecules, and the subsequent availability of increasing quantities of molecules with characterized properties, has triggered the use of statistical and machine learning approaches to automatically *learn* the structure-property relationship from these pools of characterized molecules.

Decades of research in machine learning and statistics have provided a wide range of methods for that purpose, ranging from classical least-square linear regression to artificial neural networks or decision trees [1]. Each method has is specific strengths and weaknesses, but a common issue when one wants to infer a structure-property relationship is represented by the representation of the molecular structure. Small molecules are often represented as 2D or 3D structures in chemistry and chemoinformatics, but most statistical methods, including linear models and nonlinear neural networks, require vectors an input. Molecules must therefore be first encapsulated as finite-dimensional vectors, using various molecular descriptors, before being presented as input to these algorithms. However, the construction of molecular descriptors is a difficult task. Often a significant chemical expertise coupled with heuristic feature selection methods is needed to select, among the plethora of possible molecular descriptors, the most relevant ones for a property to be predicted. The number of molecular descriptors must moreover be kept as small as possible to limit the complexity of the inference task.

The alternative to this issue is to explicitly consider molecules as *structured objects*, which, in practice, usually represents molecules by graphs. Several structure-based approaches to virtual screening have been proposed, including notably similarity-based algorithms [2], artificial neural networks based on molecular graphs [3] or graph-machine algorithms [4]. Another possibility has emerged recently with the advent of support vector machines (SVM) and related kernel methods in machine learning [5-7]. SVM is an algorithm for pattern recognition and regression that provides a useful framework to overcome the difficulty of data representations as vectors of low dimensions. From a theoretical point of view, SVM are able to infer models in large or even infinite dimensions from a finite number of observations, because the complexity of the learning task is not directly related to the dimension of the input vectors, but rather to some measure of complexity of the classification rules which are precisely controlled by SVM through the use of regularization [5]. From a practical point of view, a computational trick known as the *kernel trick* allows the estimation of models with a complexity that does not depend on the dimension of the input, but only on the number of training points. Combined together, these properties give SVM the ability to work with molecules represented by vectors of large dimension in a computationally efficient framework,

*Address correspondence to this author at the Centre for Computational Biology, Ecole des Mines de Paris, ParisTech, 35 rue Saint Honoré, 77305, Ontainebleau, France; Tel = +33-1-5624-6917;
E-mail: Jean-Philippe.Vert@mines-paristech.fr

1386-2073/09 $55.00+.00

leveraging the burden of feature selection and giving the modelers new opportunities to imagine large sets of molecular descriptors.

SVM often provide state-of-the-art performances on many classification and regression tasks, and enjoy therefore an increasing popularity in various application fields, including bioinformatics and chemoinformatics [8]. For example, SVM have been applied to the prediction of the activity of molecules on a number of target classes [9-15], toxicological properties [16-18], drug-likeliness [19-21], blood-brain barrier permeability [22], enantioselectivity [23], aqueous solubility [24], or isoelectric point [25], to name just a few. We refer the interested reader to [26] for a thorough review of SVM applications in chemistry.

While most recent successful applications of SVM in chemoinformatics were obtained by just plugging classical molecular descriptors to the SVM, an increasing line of work seeks to investigate the unique opportunities offered by SVM to go beyond classical fingerprints and molecular descriptors, based on the kernel trick. This avenue was pioneered simultaneously and independently by Kashima *et al.* [27] and Gärtner *et al.* [28] who proposed to represent the 2D structure of a molecule by an infinite-dimensional vector of linear fragment counts and showed how SVM can handle this representation with the kernel trick. Later work quickly refined these 2D kernels [29-31] and proposed new infinite-dimensional representations of 3D structures [32-34].

These first attempts to enlarge the flexibility of molecular descriptor-based predictive models represent a promising direction for *in silico* modeling of structure-property relationship, because they illustrate the unique possibilities offered by SVM and more generally, by kernel methods in this context. In this paper we review molecular kernels in SVM with the hope to offer a state-of-the-art description of the latest development in this field, and to present an invitation for the chemoinformatics community to further investigate these possibilities. For that purpose we first provide a quick introduction to SVM and kernels in Section 1, and illustrate the relevance of the kernel trick when working with 2D structures of molecules with a simple example of 2D kernel in Section 2. This example is further generalized and connected to recent work on 2D kernels in Section 3, and practical issues with these kernels are discussed in Section 4. In Section 5 we present another approach that focuses on the representation of 3D structures of molecules, and discuss practical issues for this approach in Section 6. We conclude by a discussion and suggestions for future work in Section 7.

1. Support Vector Machines and Kernels

SVM is a machine learning algorithm for pattern recognition originally developed in the early 1990's by V. Vapnik and coworkers [5,35]. Although various extensions to multi-class classification, regression, outlier detection or feature construction also exist, we focus in this review on the simple pattern recognition problem and refer the interested reader to various textbooks to know more about these extensions, collectively known as *kernel methods* [6,7]. A pattern recognition problem occurs when one is given a finite set of objects that belong to two possible classes, and must *learn* from this training set a rule to automatically *predict* the class of objects with unknown class. This general and abstract formulation encompasses in fact a number of practical situations in chemoinformatics and beyond. We focus here in particular on situations where the objects available are small molecules, and the classes to be predicted represents various properties of interest such as toxic/nontoxic, druggable/non-druggable, or inhibitor/non-inhibitor of a given target. Hence a typical pattern recognition problem could be, given a list of toxic and non-toxic molecules, to learn a rule to predict whether a new candidate molecule is toxic or not.

More formally, we represent the training set available as a set of *n* objects $x_1,...,x_n \in \mathcal{X}$, where $\mathcal{X}$ denotes the set of all possible objects, and associated binary labels $y_1,...,y_n \in \{-1\,;\,1\}$. In our case each object x_i represents a molecule, $\mathcal{X}$ denotes the set of all possible molecules, and the two classes 1 and -1 are arbitrary representations of two classes or interest, such as "toxic" and "non-toxic". Pattern recognition algorithms, such as SVM, use this training set to produce a classifier $f : \mathcal{X} \mapsto \{-1\,;\,1\}$ that can be used to predict the class of any new data $x \in \mathcal{X}$ by the value $f(x)$. We first consider the simple case where the data to be processed are represented as vectors, and we want to infer a linear classification function. We will see in the next subsection how this apparently restricted framework can be easily extended to more complex situations with the use of kernels. When objects are *d*-dimensional vectors, that is, when $\mathcal{X} = \mathbb{R}^d$, the classifier output by SVM is based on the sign of a linear function:

$$f(x) = \text{sign}\left(\langle w, x\rangle + b\right), \tag{1}$$

for some $(w,b) \in \mathcal{X} \times \mathbb{R}$ defined below. In this case the classifier has a geometric interpretation: the hyperplane $\langle w, x\rangle + b = 0$ separates the input space $\mathcal{X}$ into two half-spaces, and the prediction of the class of a new point depends on its position on the one or on the other side of the hyperplane. The particular hyperplane selected by SVM is the one that solves the following optimization problem:

$$\min_{w,b}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \mathrm{L}\left(y_i, \langle w, x_i\rangle + b\right)\right\}, \tag{2}$$

where C is a parameter and $\mathrm{L}(y,t)$ is the *hinge loss* function equal to 0 if $yt > 1$, and $1 - yt$ otherwise. For a given training example (x_i, y_i), the hinge loss term $\mathrm{L}\left(y_i, \langle w, x_i\rangle + b\right)$ quantifies how "good" the prediction $\langle w, x_i\rangle + b$ of a candidate classifier (w,b) is, in the sense that the better the prediction, the smaller the loss. For example there is no loss when w and b are such that $y_i\left(\langle w, x_i\rangle + b\right) \geq 1$, which means that $\langle w, x_i\rangle + b$ has the sign of y_i and is larger than 1 in absolute value. In other words the loss is zero when the prediction is correct and made with large confidence. Now the second term in the sum (2) is the average loss over the training set of the candidate classifier (w,b): it is small when the classifier fits well the

training points, *i.e.*, makes on average "good" predictions. On the other hand, the classifier has a large margin when the first term $\|w\|^2$ in (2) is small when the slope of the classifier is small. The two terms in (2) are conflicts, especially for high dimensional data often difficult to fit the training points well with linear functions of limited slopes. The rational behind the optimization problem (2) is indeed to find a linear classifier that reaches a trade-off between the goodness of fit on the training set (as quantified by the second term of this sum), and the smoothness of the classifier (as quantified by the first term). The parameter C controls this trade-off, by balancing the importance of each term. In the extreme case when $C = +\infty$ and the training points can be correctly separated by a hyperplane, then no error is allowed on the training set and the classifier with largest margin is found, as shown in Fig. (**1**).

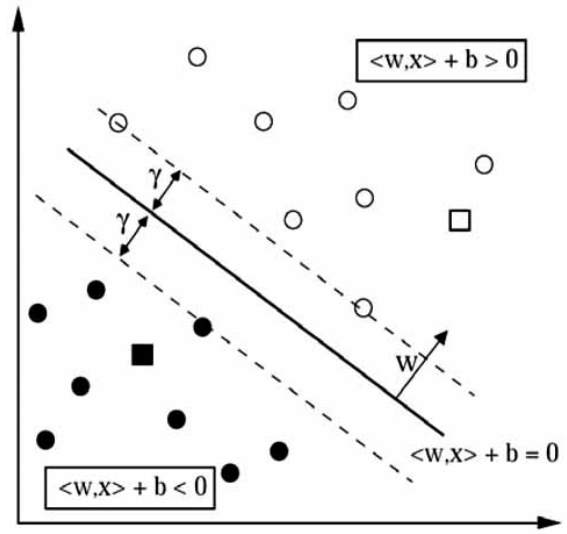

Fig. (1). SVM estimates a linear separation between the classes. When the training patterns are linearly separable and the trade-off parameter C in Equation (2) is set to $+\infty$, then the separating hyperplane selected by SVM is the *maximum margin hyperplane*, that is, the one that maximizes the distance to the closest point on each side (γ on this picture). In general, some training points may be misclassified by the selected hyperplane to control overfitting.

It is often interesting to rewrite problem (2) in an equivalent way, using classical optimization theory. Indeed, this problem is equivalent to the following quadratic problem, called its dual:

$$\max_{\alpha\in\mathbb{R}^n}\left\{\sum_{i=1}^{n}\alpha_i-\frac{1}{4}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\langle x_i,x_j\rangle\right\},$$
$$\text{subject to: } \sum_{i=1}^{n}\alpha_i=1 \text{ and } 0\le\alpha_i\le C,\ i\in\left[1:n\right]. \tag{3}$$

Both problems (2) and (3) are equivalent in the sense that the solution (w^*, b^*) of the primal problem (2) can be deduced from the solution α^* of the dual problem (3). In particular, it can be shown that $w^* = \sum_{i=1}^{n}\alpha_i^* y_i x_i$, and b^* can also be deduced from α^*. As a result, the decision function (1) can also be expressed in terms of the solution α^* of the dual problem:

$$f(x)=\text{sign}\left(\sum_{i=1}^{n}\alpha_i^*\langle x,x_i\rangle+b^*\right). \tag{4}$$

Let us now consider the use SVM for pattern recognition with molecules, represented for example by their 2D or 3D structures. Such structures being not vectors, they can not be directly input to SVM. Instead we need to embed the set of 2D or 3D structures of molecules X to a vector space H through a mapping $\phi:\mathsf{X}\mapsto\mathsf{H}$. We can then apply the SVM algorithm to the training vectors $\phi(x_i)$, $i=1,...,n$, as illustrated in Fig. (**2**). An important point to notice is that in the dual formulation (3), the data are only present through dot-products: pairwise dot-products between the training points during the learning phase in (3), and dot-products between a new data and the training points during the prediction phase in (4). This means that instead of explicitly knowing $\phi(x)$ for any $x\in\mathsf{X}$, it suffices to be able to compute inner products of the form:

$$k(x,x')=\langle\phi(x),\phi(x')\rangle, \tag{5}$$

for any $x, x'\in\mathsf{X}$. In that case the dual optimization problem (3) solved by SVM can be rewritten as follows:

$$\max_{\alpha\in\mathbb{R}^n}\left\{\sum_{i=1}^{n}\alpha_i-\frac{1}{4}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k(x_i,x_j)\right\},$$
$$\text{subject to: } \sum_{i=1}^{n}\alpha_i=1 \text{ and } 0\le\alpha_i\le C,\ i\in\left[1:n\right]. \tag{6}$$

Moreover the classification function (4) becomes:

$$f(x)=\text{sign}\left(\sum_{i=1}^{n}\alpha_i^* k(x,x_i)+b^*\right). \tag{7}$$

Hence we see that for both the training of the SVM (6) and the prediction of the class of new points (7), the feature map ϕ only appears through the function k, which is called a *kernel*. Importantly it is sometimes easier to compute directly the kernel $k(x,x')$ between two points than their explicit representations as vectors in H. In fact a classical result of Aronszajn [36] characterizes all functions $k:\mathsf{X}\times\mathsf{X}\mapsto\mathbb{R}$ that are valid kernel, *i.e.*, for which there exists a feature space H and a mapping $\phi:\mathsf{X}\mapsto\mathsf{H}$ such that (5) holds (they constitute the so-called class of *positive definite* functions). Hence, with this characterization at hand, any kernel k can be used with a SVM as long as it satisfies the positive definiteness property. This idea to replace the inner product by a positive definite kernel is often referred to as the *kernel trick*. The practical consequence of this trick is that it offers the possibility to perform the computations in the input space X, alleviating the need for the explicit computation of the vector representations $\phi(x)$.

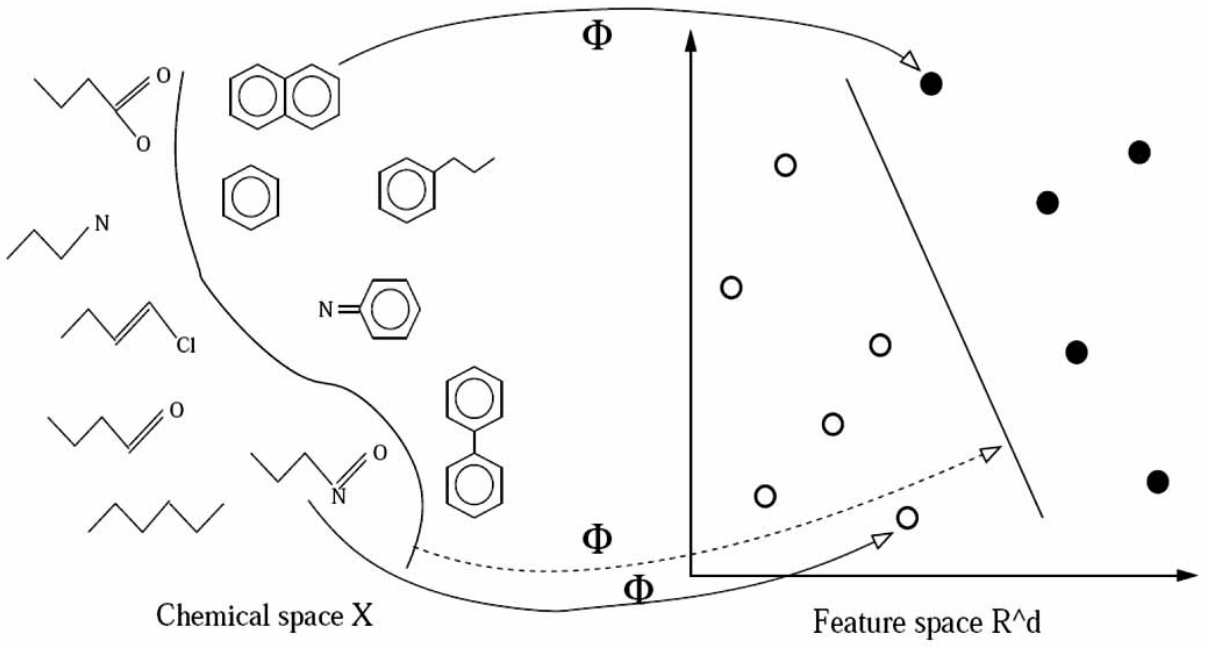

Fig. (2). In order to use SVM with molecules, we need to define an embedding of the space of molecules to a feature space ($\mathbb{R}^d$ in this picture) in which a molecule x can be represented as a vector $\phi(x)$. Note that, contrary to usual fingerprint-based approaches, the feature space might have a large or even infinite dimension.

The formulation of SVM in terms of kernels (6-7) offers at least two major advantages over the formulation in terms of explicit vectors (3-4). First, it enables the straightforward extension of the linear SVM to non linear decision functions by using a nonlinear kernel, while keeping its nice properties intact (*e.g.*, unicity of the solution, robustness to over-fitting, etc...). As an example, the Gaussian kernel $k(x,x') = \exp\left(-\|x-x'\|^2 / 2\sigma^2\right)$ is positive definite and can therefore be used as a kernel in the SVM algorithm (6). Plugging this kernel into (7) we see that the resulting discrimination function has the form:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* \exp\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right) + b^*\right),$$

which is clearly a nonlinear function of x. Second, this formulation offers the possibility to directly apply SVM to non-vectorial data provided a positive definite kernel function to compare these structures is defined. In particular, SVM have recently been successfully applied in the fields of bioinformatics and natural language processing with the introduction of kernels for sequences and trees, that find many natural applications for the analysis of DNA sequences, protein primary structures and glycans on the one hand [8,37], and text and parse trees on the other hand [38,39]. Ushering in this avenue, recent years saw the development of a whole family of kernels able to deal with 2D and 3D structures of molecules. The definition of such *structure kernels* for molecules is explained in the following sections.

2. A Simple Kernel for 2D Structures

It is common to describe the 2D structure of a molecule as a labeled undirected graph $G = (V, E)$, with atoms as vertices V and covalent bonds as edges E [40,41]. Here we assume that a label is assigned to each node and edge, typically to describe the type of atoms and bounds involved. In order to train linear models for structure-property relationship prediction, each labeled graph G representing a molecule must first be transformed into a vector $\phi(G)$. In this section we describe a simple vector representation obtained by counting all walks of a given length n, and show the relevance of the kernel formulation in this case.

A walk of length n on a graph $G = (V, E)$ is a sequence of n adjacent vertices, that is, a sequence $(v_1, ..., v_n) \in V^n$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, ..., n-1$. We note that this definition allows a given vertex or edge to be present more than once in a walk. Clearly, the number of walks of length n on a graph G is finite, and we denote with $W_n(G)$ this set of walks in the following. By concatenating the labels of the vertices and of the edges of a walk w we obtain a sequence of labels which we denote by $l(w)$, the label of the walk w. Moreover, we note L_n the set of possible labels for walks of length n, *i.e.*, all possible sequences alternating n vertex labels with $n-1$ edge labels. Fig. (**3**) illustrates these definitions. Now a simple way to represent a graph G by a vector is to extract all walks of length n from its structure, sort them by label, and count in $\phi(G)$ the

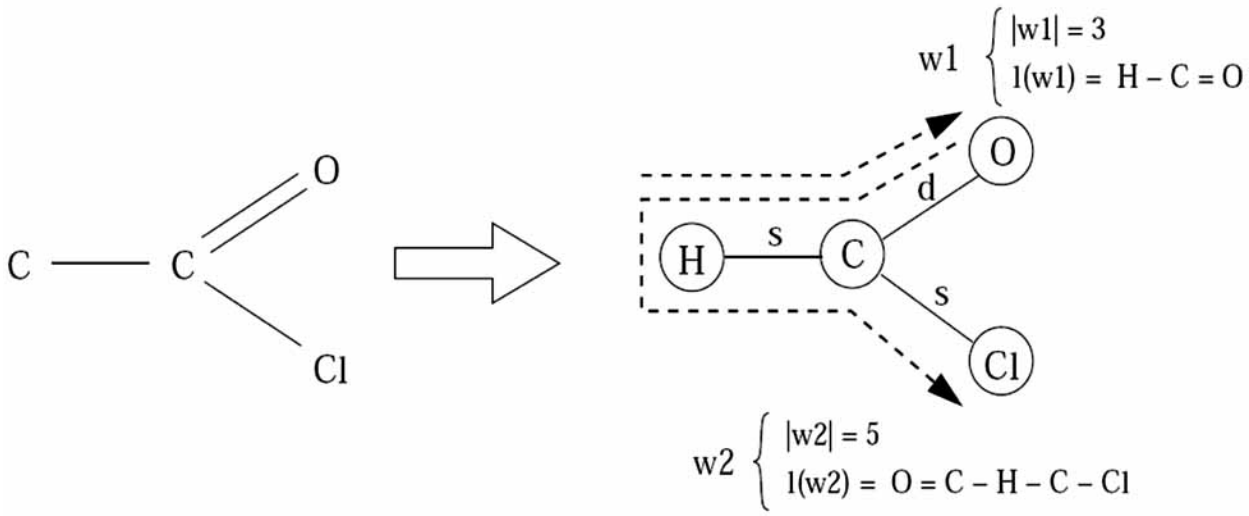

Fig. (3). The 2D structure of a molecule (on the left) can be represented by a labeled graph (on the right). Two walks on the graph are shown, together with their label and length.

number of walks with each possible label in L_n. In other words the dimension of $\phi(G)$ is equal to the size of L_n, and for each possible walk label $l \in L_n$ we define the coordinate $\phi_l(G)$ as the number of walks in G having label l. More formally the feature $\phi_l(G)$ is defined by:

$$\phi_l(G) = \sum_{w \in W_n(G)} \mathbf{1}\big(l(w) = l\big), \tag{8}$$

A direct approach to train a linear model with these vector representations would require the explicit computation and storage of $\phi(G)$ for all graphs G in a dataset. This approach becomes problematic when n becomes large, because the number of walk labels increases exponentially with n. As an example, keeping only 6 types of atoms and 3 types of covalent bonds, the number of possible labels reaches 1,944 for walks of length 3; 34,992 for walks of length 4; 629,856 for walks of length 5; and more than 3 billions for walks of length 8. This explosion in the dimension of $\phi(G)$ suggests in practice either to restrict oneself to walks of length 2 or 3, or to compress the representation $\phi(G)$. The later approach is widely used in chemoinformatics because fragments of length $5-10$ are known to provide useful information in many structure-property relationship problems. The solution most often encountered is to use a hash table of limited size (typically 1024 or 2048) to map the vector $\phi(G)$ onto a vector of smaller dimension, called a molecular fingerprint [41]. An obvious drawback of this solution is the danger of clashes, *i.e.*, the mapping of different labels to the same position in the hashed vector.

An alternative solution for the use of large n values is to use kernels. As we now show, indeed, kernels allow the estimation of linear models for vectors $\phi(G)$ without reducing their dimension nor requiring the computation and storage of the vectors. Indeed, remembering from Section 1 that SVM only need the definition of the inner product between vectors to estimate a linear problem, we only need to show how the inner product for the vector representation (8) can be computed efficiently. For that purpose, let us write this inner product more explicitly for any two graphs G and G':

$$\begin{aligned}
\langle \phi(G), \phi(G') \rangle &= \sum_{l \in L_n} \phi_l(G)\phi_l(G') \\
&= \sum_{l \in L_n} \left(\sum_{w \in W_n(G)} \mathbf{1}\big(l(w) = l\big) \right) \left(\sum_{w' \in W_n(G')} \mathbf{1}\big(l(w') = l\big) \right) \\
&= \sum_{w \in W_n(G)} \sum_{w' \in W_n(G')} \left(\sum_{l \in L_n} \mathbf{1}\big(l(w) = l\big)\mathbf{1}\big(l(w') = l\big) \right) \\
&= \sum_{w \in W_n(G)} \sum_{w' \in W_n(G')} \mathbf{1}\big(l(w) = l(w')\big).
\end{aligned} \tag{9}$$

In other words the inner product between $\phi(G)$ and $\phi(G')$ can be expressed exactly as the number of pairs of walks (w, w') of length n, respectively in G and G', with the same label. In order to show how this number can be computed efficiently, it is useful to introduce the *product graph* $G \times G'$ which is a graph whose vertices are pairs of vertices of G and G' with the same label, and whose edges connect pairs of vertices which are connected both in G and G' (Fig. (**4**)). In other words the vertices of $G \times G'$ are the pairs $(v, v') \in V \times V'$ with $l(v) = l(v')$, and there is an edge between (v_1, v_1') and (v_2, v_2') if and only if there is both an edge between v_1 and v_2 in G and an edge between v_1' and v_2' in G', and if both edges have the same label. It is easy to see, then, that a walk in the product graph is a sequence of pairs of vertices (v, v'), in G and G', that are connected in $G \times G'$ and therefore in G and G'. Moreover both se-

quences of vertices in G and G' are made of pairs of vertices and pairs of edges with the same label, *i.e.*, they form a pair of walks in G and G' with the same label. Conversely, given any walks w in G and w' in G' with same label $l(w) = l(w')$, there is a walk in the product graph that corresponds to the pair of walks (w, w'). In other words, there is a bijection between the pairs of walks in G and G' with the same label, on the one hand, and the walks on $G \times G'$, on the other hand. Hence counting the number of pairs of walks of length n on G and G' with the same label is equivalent to simply counting the number of walks of length n on $G \times G'$, as shown in Fig. (**4**). It turns out that counting the number of walks of length n on a general graph (and in particular on a product graph for our purpose) can be easily computed by a recursion over n. Indeed, for a general graph, if we denote by $A_i(v)$ the number of walks of length i starting at vertex v, then $A_i(v) = 1$ for any vertex v and the following recursion formula holds:

$$A_{i+1}(v) = \sum_{u \sim v} A_i(u), \tag{10}$$

where the sum is over the neighbor vertices of v. $A_n(u)$ can therefore be computed for any $u \in V$ by applying this formula recursively over i. The number of walks of length n on the graph is then simply obtained by summing $A_n(u)$ over the vertices u. We observe that if we denote by $\boldsymbol{A}$ the adjacency matrix of the graph and by $\mathbf{1}$ the vector whose entries are all equal to 1, then (10) simply expresses $\boldsymbol{A}^{i+1}\mathbf{1}$ as $\boldsymbol{A} \times \boldsymbol{A}^i \mathbf{1}$, and the count of walks of length n is equal to $\mathbf{1}^T \boldsymbol{A}^{n-1} \mathbf{1}$.

To summarize, we have shown that for the vector representation (8), the inner product between two graphs G and G' representing the 2D structures of two molecules can be computed by (i) constructing the adjacency matrix $\boldsymbol{A}$ of the product graph $G \times G'$ and (ii) computing $\mathbf{1}^T \boldsymbol{A}^{n-1} \mathbf{1}$ using the recursion (10). This computation is exact and efficient, although the dimension of the vectors can reach the billions. In particular, the complexity of the computation increases only linearly with n, while the number of features increases exponentially. Using this inner product with a kernel method for pattern recognition or regression allows to estimate a linear model in this space without ever computing nor storing any vector.

3. 2D Kernel Extensions

The kernel for 2D structures presented in the previous section to illustrate the power of kernels can be used as such, but many extensions have been proposed to increase the flexibility and the expressiveness of the representation. In this section we review some of these extensions.

3.1. Walks of Various Lengths

In the computation of the kernel based on walks of length n, we note that kernels of length $i < n$ are computed as intermediaries. The choice of n is arbitrary in practice and should depend on the targeted application and the data available. Alternatively we may decide not to choose a particular value of n, but to combine walks of different lengths in a joint feature model. The inner product being additive when new features are added, the kernel corresponding to the feature space (8) where all walks of length up to n are considered is the sum of the kernels corresponding walks of fixed length smaller than n. The complexity of the computation is barely increased for this extension: instead of performing the recursion (10) n times before summing the terms, one just needs to increase a counter by the sum of the terms at each iteration.

When n increases the inner product in this "until-n" extension grows exponentially with n and diverges. In order to use large values for n, and even infinite n to include every single walk in the kernel, one has to weight the contribution of the different walks by a factor $\lambda(w)$ that will ensure the convergence of the series. This leads to the following kernel definition:

$$k(G, G') = \sum_{n=1}^{\infty} \sum_{w \in W_n(G)} \sum_{w' \in W_n(G')} \lambda(w)\lambda(w') \mathbf{1}\Big(l(w) = l(w')\Big) \tag{11}$$

As an example, Gärtner [42] proposed to weight the contribution of walks of length i in the inner product by a factor $\beta^{i/2}$, *i.e.*, to consider the formula:

$$\begin{aligned} k(G, G') &= \sum_{n=1}^{\infty} \sum_{w \in W_n(G)} \sum_{w' \in W_n(G')} \beta^n \mathbf{1}\Big(l(w) = l(w')\Big) \\ &= \sum_{n=1}^{\infty} \beta^n k_n(G, G'), \end{aligned} \tag{12}$$

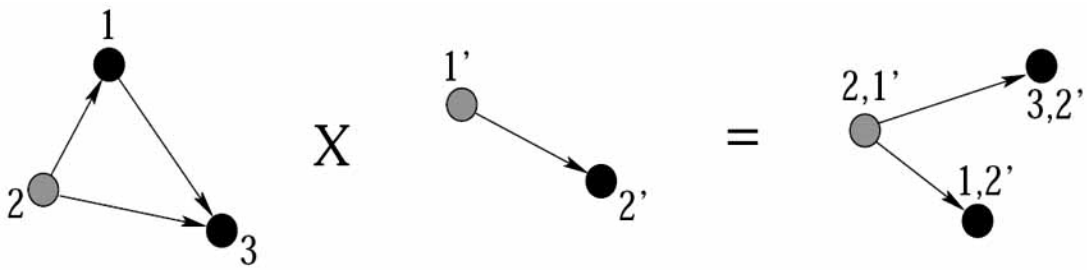

Fig. (4). The product graph of two graphs (on the left) is obtained by considering all pairs of vertices with similar labels as vertices, and connecting two such vertices when the respective pairs of vertices in the initial graphs are connected (on the right). Each walk in the product graph (*e.g.*, $(2,1') - (3,2')$) is associated to a pair of walks in the initial graphs with same labels (*e.g.*, $2-3$ and $1'-2'$), and vice-versa.

where k_n denotes the kernel based on the count of walks of length exactly n. Remembering from the previous Section that $k_n(G,G')$ is equal to $\mathbf{1}^T \boldsymbol{A}^{n-1}\mathbf{1}$, where $\boldsymbol{A}$ is the adjacency matrix of $G \times G'$, we can rewrite and factorize this kernel as follows for β small enough:

$$\begin{aligned} k(G,G') &= \sum_{n=1}^{\infty} \beta^n \mathbf{1}^T \boldsymbol{A}^{n-1}\mathbf{1} \\ &= \beta \mathbf{1}^T \left(\sum_{n=0}^{\infty} \beta^n \boldsymbol{A}^n \right) \mathbf{1} \\ &= \beta \mathbf{1}^T \left(\mathbf{I} - \beta \boldsymbol{A} \right)^{-1} \mathbf{1} . \end{aligned} \quad (13)$$

Hence the computation of the inner product in the infinite-dimensional space of all walk counts can be performed explicitly, at the cost of inverting the sparse matrix $\mathbf{I} - \beta A$ (where $\mathbf{I}$ denotes the identity matrix). In practice the first terms of the power series expansion provide a fast and good approximation to the complete kernel, and allow more flexibility in the weighting of the walks of different length.

Another weighting scheme for walks has been proposed independently by Kashima *et al.* [27], who propose to define Markov random walks of each graph and weight the occurrence of each walk on a graph by its probability under the corresponding random walk model. As for the exponential decay, the random walk weighting scheme factorizes along the walks and can be computed with the same tricks as the exponential decay walk kernel.

3.2. Filtering Tottering Walks

In the previous section, we did not make any restriction on the definition of walks: they are simply defined as successions of connected graph vertices. Because molecular graphs are essentially undirected, this generic definition allows walks to have an erratic behavior, which can lead in turn to a misleading information about the true structure of the graph in the kernel. Indeed, arbitrarily long walks can for instance be generated by simply alternating between two connected vertices. A natural way to increase the expressive power of walks with respect to the structure of the graphs is to prevent vertices from appearing more than once in a walk. In the terminology of graph theory, this corresponds to defining a kernel based on common *paths* instead of common walks. Albeit very natural, this extension unfortunately renders the kernel computation untractable [28].

A computationally efficient alternative proposed by Mahé *et al*. [30] is to disregard the *tottering walks* in the enumeration of walks. As illustrated in Fig. (**5**), a tottering walk is a walk that goes from a vertex i to a vertex j and back to vertex i. Although the notion of path is stronger than the notion of tottering walks for general graphs, they are equivalent on graphs without cycles. The relevance of the concept of tottering walks stems from computational advantages. Indeed, it can be shown that the set of tottering walks of a graph G corresponds to a set of walks of a transformed graph $t(G)$, where the transformation t involves adding additional vertices and edges. As a result, the kernel for two graphs G and G' based on non-tottering walks only is easily computed as the standard walk kernel between the transformed graphs $t(G)$ and $t(G')$. More details about this transformation can be found in [30].

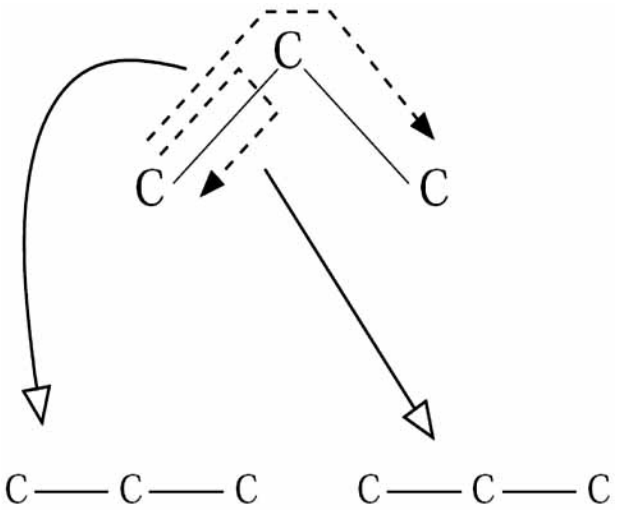

Fig. (5). Tottering (right) and no-tottering (left) walks. These two walks are labeled as a succession of 3 carbon atoms, but only the no-tottering one involves 3 distinct atoms.

3.3. Increasing the Expressiveness of Walks

A second criticism that can be made to walk kernels is the fact that, because of their linearity, walks bear limited information about the structure of a graph. A principled way to address this issue, which is actually the topic of the next subsection, is to introduce subgraphs of a higher level of complexity in the kernel construction. In practice, however, this approach usually raises additional complexity issues that can be hard to circumvent. A simpler alternative is to keep a walk-based characterization of graphs and introduce some form of prior knowledge in the graph labeling function, in order to enrich the information brought by walks about the graph structure. This is in particular the approach taken in Mahé *et al.* [30] where a new set of labels is defined for the vertices of a graph, based on the local environment of the atoms in the corresponding molecule. This method relies on a topological index, called – with a slight abuse – the *Morgan index*, defined for each atom of the molecule according to the following iterative procedure, borrowed from the well known Morgan algorithm used for the canonical numbering of molecular structures [43]. Initially, the index associated to every vertex is equal to 1. Then, at each step, the index of a vertex is defined as the sum of the indices associated to its neighbors at the previous iteration. This process is straightforward to implement in practice, since if we let M_i be the vector of Morgan indices computed at the i-th iteration, it reads as $M_0 = \mathbf{1}$ and $M_{i+1} = \boldsymbol{A} M_i$, where $\mathbf{1}$ is the unity vector and $\boldsymbol{A}$ is the adjacency matrix of the graph.

As shown in Fig. (**6**), Morgan indices can make it possible to distinguish between atoms having the same type but different topological properties. When they are included in

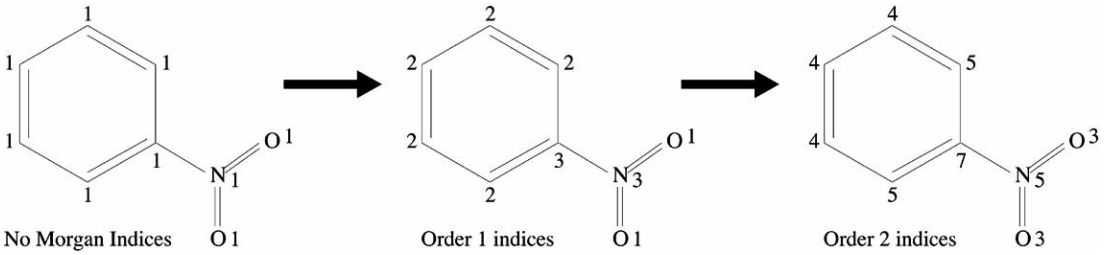

Fig. (6). Illustration of the Morgan process. Initially, all atoms of the cycle are seen as identical. For increasing iterations, the presence of the NO_2 branch is more and more reflected in the atoms of the cycle.

the labels of the vertices, these indices therefore define a walk as a sequence of atoms taken in a particular topological configuration. In practice, the advantage of this refinement is twofold. First, the introduction of topological information in walk labels enriches the information they bear with respect to the structure of the graphs to be compared. Second, because atoms are made more specific to the graph they belong to (Fig. (**6**)), the number of identically labeled atoms found in a pair of graphs automatically decreases, which has the effect of reducing the size of their product graph, hence the time of computing the kernel. Note that this computation advantage is surprisingly due to the increase in dimension of the feature space. We note however that while this Morgan process systematically reduces the cost of computing the kernel, performing too many iterations makes it impossible to detect common walks within a pair of graphs.

3.4. Subtree Kernels

As mentioned in the previous subsection, the linear nature of walks limits their ability to properly encode the structure of a graph. In particular, it is well known that graphs can be structurally different, yet have the same walk content, as illustrated in Fig. (**7**). As a result, simple graph kernels based on the count of common walks cannot distinguish between such pairs of graphs. This fact was pointed out by Ramon and Gärtner [44], who also show that computing a perfect graph kernel, that is, a kernel mapping non-isomorphic graphs to distinct points in the feature space, is at least as hard as solving the graph isomorphism problem for which there is no known polynomial-time algorithm. This therefore suggests that the expressiveness of graph kernels must be traded for their computational complexity.

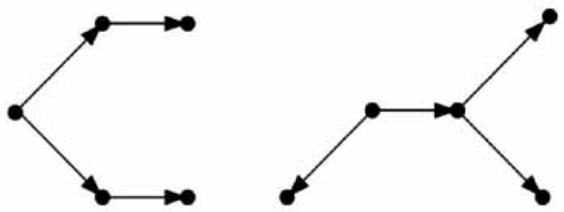

Fig. (7). Two graphs having the same walk content, namely • : ×5 ; •→• : ×4 and •→•→• : ×2, and consequently mapped to the same point of the feature space associated to the kernel based on the count of walks [28].

As a first step towards a refinement of the feature space used in walk-based graph kernels, Ramon and Gärtner [44] introduce a kernel function comparing graphs on the basis of their common subtrees. As shown in Fig. (**8**), this representation looks particularly promising for molecules, since it allows to capture in a principled way a wide range of func-

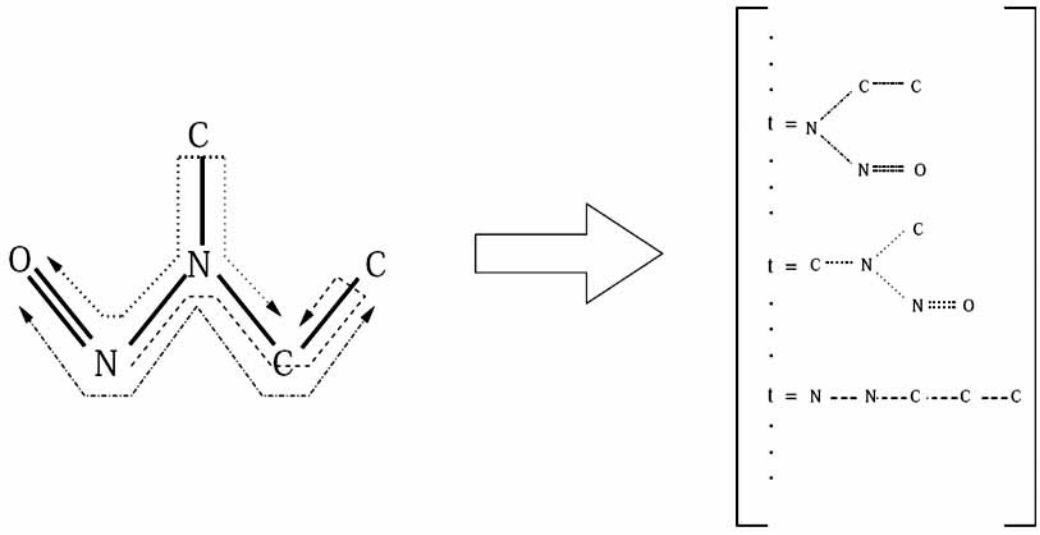

Fig. (8). Illustration of the tree-structured fragment representation: a graph G (left) and a sample of its feature space representation $\phi(G)$ (right). Note that the lowermost tree corresponds to a walk structure.

tional features of molecules, that typically correspond to specific branching patterns on their associated graphs. On the practical side, this type of kernels can be computed by means of dynamic programming algorithms that recursively detect and extend identical neighborhood properties within the vertices of the graphs to be compared, in order to explicitly build their set of common subtrees. The relative contribution of subtrees of different sizes is typically controlled by means of a parameter playing a similar role to that of the parameter β in Equation (12). These algorithms have a prohibitive complexity in general, but they can be deployed for molecular graphs where the degree of the vertices is small in average because of valence rules. The relevance of this class of kernels, as well as its relationship with standard walk-based kernels, has been analyzed in details by Mahé and Vert [45].

4. 2D kernels in Practice

As a conclusion about kernels for 2D structures, we now discuss several issues related to their application in practice.

4.1. Implementation and Complexity Issues

As mentioned in Section 2 an elegant way to compute walk based kernels lies in the product graph formalism initially introduced by Gärtner *et al.* [28]. The basic idea of the product graph construction is to merge the pair of graphs to be compared into a single graph, in such a way that a bijection is defined between the set of walks of the product graph and the set of common walks of the two initial graphs. It then follows that the number of walks of a given length occurring at the same time in the two graphs can be obtained by simple matrix products, which actually offers a closed form solution to the computation of kernels based on walks of infinite length for well chosen walk weighting schemes [28,29]. As a result, even though the dimensionality associated to these kernels can be very large, and actually infinite, computing these kernels under the product graph formalism has a polynomial complexity with respect to the product of the size of the graphs to be compared (more precisely, the worst case complexity is cubic with respect to the product of the size of the graphs).

In practice, this type of product-graph implementations remains time consuming, even for relatively small graphs, which questions the suitability of these kernels for virtual screening applications that typically involve large datasets of molecules. However, if only walks up to a given length are considered, which usually makes sense for real world applications, fast algorithms can be used to compute walk kernels, based for instance on trie-tree structures and string kernel algorithms [7,46], or standard depth-first search procedures [31]. Moreover, alternative implementations allowing to drastically reduce the time needed to compute such kernels in their general form have recently been proposed [47].

4.2. Kernel Normalization

A potential drawback of kernels comparing structured objects by means of their substructures lies in the fact that kernel values are highly dependent on the size of the objects to be compared. Indeed, big objects tend to be granted a higher degree of similarity than small objects for the only reason that they are made of a larger number of substructures. This fact can lead to a serious bias of the subsequent prediction model, and the classical way to tackle this issue it to apply a normalization operation in order to take into account the size of the objects in the value of the kernel function. In practice, the mainstream normalization scheme is given by the following expression:

$$\tilde{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}},$$

where k is the original kernel, and $\tilde{k}$ its normalized version. Note that this normalization operation has the effect of setting the diagonal of the kernel matrix to one, meaning that individual objects are given the same degree of self-similarity, whatever their size is. Geometrically, this amounts to scaling all vectors to unit norm before taking their inner product.

In the context of molecular graph kernels, alternative normalization schemes based on the Tanimoto similarity coefficient have recently been introduced [31,32]. The Tanimoto coefficient is widely used in chemoinformatics to assess the similarity of molecular fingerprints. For a pair of fingerprints (A,B), it is defined as:

$$T_{AB} = \frac{A^T B}{A^T A + B^T B - A^T B}.$$

For binary fingerprints, it can be seen as the ratio between their intersection, that is, the number of bits set to one in both fingerprints, and their union. As pointed out by Ralaivola *et al.* [31], since it is based on inner product operations, this coefficient can be generalized to any kernel function, leading to the notion of *Tanimoto kernel*, defined for a kernel k as:

$$\tilde{k}(x,y) = \frac{k(x,y)}{k(x,x) + k(y,y) - k(x,y)}.$$

This transformation provides an alternative way to normalize kernel functions in the sense that $\tilde{k}(x,x) = 1$ for all x. Several variations on this idea that allow to generalize the classical Tanimoto coefficient in different ways are proposed in [31,32].

4.3. Kernel Parameterization

Last but not least comes the issue of kernel parameterization. This question is of tremendous importance since a bad parameterization can seriously entail the success of the subsequent virtual screening application. First, one must choose to consider the kernel based on walks or the kernel based on subtrees. As for now, this question remains largely open since apart from the study of Mahé and Vert [45], the relevance of subtrees in graph kernels has not been studied in details. While the preliminary results presented in this study suggest that subtree kernels may indeed improve over their walk-based counterparts, they also show that this class of substructures raises additional issues, related in particular to the computational complexity of the kernels as well as the explosion in the number of subtrees found in the graphs.

Concerning the parameterization of walk kernels, the main issue concerns the length(s) of the walks to consider: either walks of a precise length, up to a maximal (but finite) length, or even up to infinite length. In practice, this question is highly dependent on the problem considered. Optimally choosing this parameter can therefore hardly be made *a priori* but involves cross-validation procedures. Although focusing on walks of a precise length can be optimal in some cases (for instance, walks of length 6 or 7 can be optimal to characterize molecules mainly made of aromatic cycles), a safe default choice is to consider walks of length up to a limited value to be taken around 8 or 10. Actually, because kernels based on an infinite number of walks require to down-weight the contribution of walks depending on their length (as in Equation (12) for instance), long walks are in practice so penalized that their individual contribution is barely taken into account in the kernel. Explicitly limiting the length of the walks to be taken into account therefore makes sense in practice. Moreover, considering a finite number of walks provides a greater flexibility in the way to control their relative contribution in the kernel, and offers the practical advantage of paving the way to the deployment of computationally cheaper algorithms, as discussed in Section 4.1. A second important issue is related to kernel normalization. Although the impact of choosing the first or the second normalization scheme introduced in Section 4.2 has not been analyzed in details, Tanimoto kernels led to good results in several validation studies [31,32,34]. Finally, one may consider further refinements such as filtering tottering walks and introducing Morgan indices. As shown in [30], Morgan indices of a limited order, typically obtained at the 2nd or 3rd iteration of the process, can indeed improve virtual screening models while reducing their computational costs (actually, this is only true for product-graph implementations. For trie-tree implementations for instance, Morgan indices have the opposite effect of increasing the cost of computing the kernel). Filtering tottering walks should be subject to caution however. Indeed, while this can indeed improve the models in some cases, it seems that the tottering phenomenon can also be helpful to detect similarity between structurally different compounds [45].

5. A 3D Pharmacophore Kernel

Motivated by the fact that the tridimensional structure of molecules have a central role in many biological mechanisms, including drug-target interactions for instance, recent attempts have been made to develop kernels for 3D structures of molecules. In this section, we introduce a class of kernels that relies on the notion of *pharmacophore* which is widely used in chemoinformatics. A pharmacophore is usually defined as a spatial arrangement of three to four atoms (more generally, pharmacophores are defined as arrangements of *groups* of atoms having particular properties, such as positive or negative polarity, high hydrophobicity, and so on) responsible for the biological activity of a drug molecule. In the following, we focus on *three-points pharmacophores* composed of three atoms, whose arrangement therefore forms a triangle in the 3D space (Fig. (**9**)), but similar ideas naturally apply to pharmacophores of different cardinalities (in particular, similar ideas were developed by Swamidass *et al.* [32] based on two-points pharmacophores, that is to say, distances between pairs of atoms). With a slight abuse we refer as pharmacophore below to *any* possible configuration of three atoms arranged as a triangle and present in a molecule, representing therefore a *putative* configuration responsible for the biological property of interest. More precisely, we consider a molecule m as a set of atoms in the 3D space, that is:

$$m = \left\{ \left(x_i, l_i \right) \in \mathbb{R}^3 \times \mathsf{L} \right\}_{i=1,\ldots,|m|},$$

where $|m|$ is the number of atoms that compose the molecule, and $(x_i, l_i) \in \mathbb{R}^3 \times \mathsf{L}$ stands for its i-th atom, x_i being its vector of (x, y, z) coordinates, and l_i its label, such as its type for instance, but more generally taken from a set L of atom labels. With these notations at hand, the set of three-points pharmacophores that can be extracted from the molecule m can be formally defined as:

$$\mathsf{P}(m) = \left\{ \left(p_1, p_2, p_3 \right) \in m^3,\ p_1 \neq p_2, p_1 \neq p_3, p_2 \neq p_3 \right\}.$$

Following our discussion of Section 2, a simple way to represent a molecule m is to extract all its pharmacophores, sort them by type, and count in a vector $\phi(m)$ the number of pharmacophores of each possible type. Clearly, the number of pharmacophores associated to a molecule is finite, but since their definition is based on the precise (x, y, z) coordinates of the atoms it is made of, or equivalently on continuous inter-atomic distances, the space of all *possible* pharmacophores is infinite. Defining such a vector representation therefore requires in practice to discretize the space of pharmacophores, which boils down to discretizing the range of inter-atomic distances into a pre-defined number of bins. Formally, if we consider n bins in the discretization, this operation defines a space of *discrete pharmacophores* $\mathsf{T} = \mathsf{L}^3 \times [1, n]^3$, where each pharmacophore corresponds to a

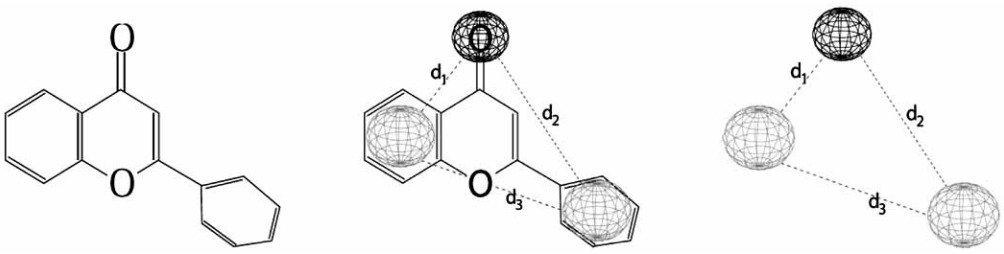

Fig. (9). Left: the molecule of flavone. Right: a pharmacophore made of one hydrogen bond acceptor (topmost sphere) and two aromatic rings, with distances d_1, d_2 and d_3 between the features that can be extracted from its structure, as shown in the middle.

triplet of atom labels, taken from the alphabet L, and a triplet of distance bin indices, taken in $[1,n]$. We can now define the vector representation $\phi(m)$, in which each coordinate $\phi_t(m)$ is the number of pharmacophores extracted from the molecule m that correspond to the discrete pharmacophore t, that is,

$$\phi_t(m) = \sum_{p \in \mathsf{P}(m)} \mathbf{1}\big(\mathrm{disc}(p) = t\big),$$

where the function $\mathbf{1}\big(\mathrm{disc}(p) = t\big)$ is one if the discretized version of the pharmacophore p is t, meaning that they are based on the same triplet of atom labels and the same triplet of distance bins, and zero otherwise. The number n of bins considered in the discretization specifies the resolution at which distinct pharmacophores are considered to be equivalent, and constitutes a critical parameterization issue. Indeed, small distance bins may prevent the detection of similar pharmacophores, while large distance bins can lead to a matching between unrelated pharmacophores. In practice, this parameter also defines the dimension of $\phi(m)$. For example, considering 6 distinct types of atoms and 10 distance bins, which corresponds to bins of $2\,\text{Å}$ if pairs of atoms are considered to lie within the $0\text{-}20\,\text{Å}$ distance range, the cardinality of T, hence the dimension of $\phi(m)$, is 216,000. This number is raised up to 1,728,000 in order to reach a precision of 1Å per interatomic distance bin. This explosion in the number of dimensions suggests again that in order to explicitly store the vector $\phi(m)$, one should either consider a limited number of bins, thereby considering a poor resolution to characterize molecular structures, or rely on hashing algorithms to map the vector $\phi(m)$ onto a vector of limited size, which as discussed previously, has the effect of inducing clashes between distinct pharmacophores. This representation highlights once again the benefit of using kernel functions since, following the lines of Equation (9), one can define the kernel:

$$\begin{aligned} k(m,m') &= \phi(G)^T \phi(G') \\ &= \sum_{p \in \mathsf{P}(m)} \sum_{p' \in \mathsf{P}(m')} \mathbf{1}\big(\mathrm{disc}(p) = \mathrm{disc}(p')\big), \end{aligned} \tag{14}$$

which, as will be discussed in Section 6.1, enables to map pairs of molecules and compute their inner product in feature spaces indexed by millions of pharmacophores, for a computational complexity that remains polynomial with respect to the product of their sizes.

Of course, the idea of representing a molecule by means of its pharmacophoric content is not new, and the above approach bears strong similarity with well known pharmacophore fingerprint representations [48-50]. The above discussion nevertheless illustrates the interest of using kernel functions in this case, since they allow to compute *exactly* the inner products between very high-dimensional feature vectors without the need of computing nor storing them, which is not possible in general and comes at the price of an information loss. This is not, however, the major improvement made possible by kernel functions in this context. Indeed, as shown in Fig. (**10**), the main drawback of this approach lies in the discretization of the pharmacophore space itself: not only the choice of the discretization step controls the precision required to match a pair of pharmacophores, but it also prevents pharmacophores falling on different sides of bins edges to be matched, although they can be very close, and actually even closer that two pharmacophores falling in the same bin. The kernel approach allows to circumvent this discretization issue by means of a simple generalization of Equation (14), where the binary function checking whether pairs of pharmacophores have the same discretized version or not is replaced by a general kernel between pharmacophores in order to continuously quantify their similarity. Letting k_P be such a kernel, this leads to the general 3D kernel formulation:

$$k(m,m') = \sum_{p \in \mathsf{P}(m)} \sum_{p' \in \mathsf{P}(m')} k_P(p,p'), \tag{15}$$

which was introduced in Mahé *et al.* [33]. A meaningful kernel k_P between pharmacophores should intuitively quantify at the same the similarity of the triplets of atoms the pair of pharmacophores to be compared are defined from, and the similarity of their spatial arrangement. A natural way to achieve this goal, which is at the same time compatible with the algorithm implementing the kernel (15) (see Section 6.1), consists in factorizing the kernel k_P along the pairs of atoms and inter-atomic distances that define the pair of pharmacophores to be compared. Mahé *et al.* [33] suggest for instance to introduce elementary kernel functions $k_{At} : \mathsf{L} \times \mathsf{L} \to \mathbb{R}$ and $k_{Dist} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ comparing atoms and distances respectively, and to define the kernel k_P as:

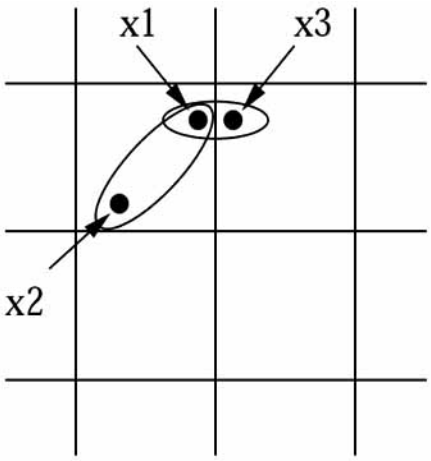

Fig. (10). Illustration of the discretization issue. In this picture, x_1, x_2 and x_3 represent pharmacophores living in a discretized bidimensional (Euclidean) space. x_1 is closer to x_3 than it is from x_2, yet the discretization affects x_1 and x_2 to the same bin and x_3 to another bin. The kernel of Equation (15) allows to circumvent this issue.

$$k_P(p,p') = \prod_{i=1}^{3} k_{At}(l_i, l_i') \prod_{i=1}^{3} k_{Dist}\left(\left\|x_i - x_{i+1}\right\|, \left\|x_i' - x_{i+1}'\right\|\right), \quad (16)$$

where the pharmacophore p (resp. p') is defined as $\left((l_i, x_i)_{i=1:3}\right)$ (resp. $\left((l_i', x_i')\right)_{i=1:3}$), $\|.\|$ denotes the Euclidean distance, and the index $i+1$ is taken modulo 3. In this approach, the task of defining a kernel between 3D structures therefore boils down to defining a couple of kernels comparing atoms and inter-atomic distances. These kernels intuitively define the elementary notions of similarity involved in the pharmacophore comparison, which in turns define the overall similarity between molecules. A simple default choice for these kernels is to define the atom kernel k_{At} as a binary kernel simply checking whether the pair of atoms to be compared has the same label or not, that is:

$$k_{At}(l,l') = \mathbf{1}(l = l'),$$

and to define the inter-atomic distance kernel k_{Dist} as the following Gaussian radial basis function (RBF) kernel:

$$k_{Dist}(x,y) = \exp\left(-\frac{\|x-y\|^2}{\theta^2}\right),$$

where σ is a bandwidth parameter. Under this parameterization, it is interesting to note that the continuous kernel of Equation (15) and its discretized counter part of Equation (14) share an important feature: because the atom kernel k_{At} is binary, both kernels are based on pairs of pharmacophores defined by the same triplets of atom labels. The striking difference between the two formulations lies in the fact that in the kernel of Equation (15), the strength of the pharmacophore matching is *continuously* controlled by the parameter σ of the (RBF) kernel comparing inter-atomic distances. Choosing a small value of σ corresponds to imposing a strong constraint on the spatial similarity of pharmacophores, while a larger value of σ allows pairs of pharmacophores to be taken into account in the kernel although their spatial configurations may differ.

We conclude this section by noting that the class of kernels defined by Equation (15) does not have an explicit inner product interpretation in general, and in particular using the above parameterization. Nevertheless, this construction is known to be valid as long as the kernel k_P is a proper kernel function [51].

6. 3D Kernels in Practice

In this Section, we discuss general considerations related to the application of 3D kernels in practice.

6.1. Implementation and Complexity Issues

Without going into technical details, it can be shown that the class of pharmacophore kernels introduced in Section 5 can be computed by algorithms derived from those used for the computation of 2D kernels. Indeed, while the 3D structure of a molecule was previously defined as a set of atoms in the 3D space, it can equivalently be seen as a fully connected labeled (and undirected) graph, with atoms as vertices and inter-atomic distances as edge labels. Under this representation, it is easy to see that computing the continuous kernel of Equation (15) can be interpreted as computing a walk kernel restricted to the walks that define cycles of length 3 on the graphs. Moreover, provided the kernel k_P factorizes along the pair of pharmacophores to be compared, which is the case of the kernel proposed in Equation (16), it is easy to show that this kernel can be computed by product-graph algorithms and simple matrix product operations, for a cubic complexity with respect to the product of the sizes of the molecules to be compared. While this complexity can be prohibitive for applications involving large datasets of molecules, the discretized version of the kernel can benefit from fast implementations derived, here also, from string kernel algorithms and trie-tree structures. We refer the interested reader to [33] for a detailed discussion about the implementation and the computational complexity of these kernels.

6.2. Kernel Parameterization

In its discretized version, the only parameter entering the definition of the kernel is the number n of bins to discretize the inter-atomic distances. As already noted in Section 5, this parameter is of critical importance since it controls the precision up to which pharmacophores are considered to be identical or not. Unfortunately, this parameter can hardly be chosen a priori, and Mahé *et al.* [33] suggest to optimize this parameter using cross-validation procedures. In this study, when optimized over the grid $\{4,6,8,...,30\}$ for a $0-20$ Å inter-atomic distance range, this parameter was usually taken between 20 and 30, which suggests that the matching between a pair of pharmacophores should be subject to strong spatial constraints. On the other hand, such fine grained resolutions have the effect of increasing the impact of the discretization issue illustrated in Figure 10.

Under the parameterization proposed in Section 5, the only parameter entering the definition of the general kernel of Equation (15) is the bandwidth σ of the RBF kernel between inter-atomic distances. In the above study, small values of σ, which correspond to strong spatial constraints in the pharmacophore comparison, are usually selected by cross-validation procedures. While in these cases the discrete and continuous formulations of the kernel tend to coincide (indeed, in the extreme case where σ tends to 0 and n to $+\infty$, both formulations are equivalent), the continuous formulation usually led to better performance in this study.

6.3. Molecule Enrichment

Many mechanisms of interest in tridimensional virtual screening involve specific physicochemical properties of the molecules. In the case of drug-target interaction for instance, the molecular mechanisms responsible for the binding are known to depend on a precise 3D complementarity between the drug and the target, from both the steric and electrostatic perspectives. For this reason, standard pharmacophore based approaches, and in particular pharmacophore fingerprints, usually define pharmacophores from atoms or groups of atoms having particular properties. Typical molecular features of interest are positive and negative charges, high hy-

drophobicity, hydrogen donors and acceptors and aromatic rings [52].

Similarly to the introduction of Morgan indices in 2D kernels discussed in Section 3.3, the atom-based kernel constructions presented in the previous section can naturally be extended to integrate this type of external information using specific label enrichment schemes. For instance, Mahé *et al.* [33] use a simple scheme where the label of an atom is composed of its type and the sign of its partial charge. Positively-charged, neutral and negatively-charged atoms of carbon are therefore labeled as $\{C^+, C^0, C^-\}$ in this approach. Alternative labeling schemes are considered in Azencott *et al.* [34], based in particular on element hybridization, where for instance an sp^3 carbon atom is labeled as $C.3$, and a typing of atoms according to conventional pharmacophoric features, such as polarity, hydrophobicity, and hydrogen-bond donors and acceptors. These studies show that, in general, such label enrichments have a positive influence on the subsequent structure-activity relationship models, while enabling to drastically reduce the computation cost of the kernels in some cases [33].

6.4. Conformational Analysis

For real-world applications, considering the tridimensional structure of molecules raises the additional issue of *conformational analysis*. Indeed, because of the presence of rotational bonds, molecules are not static in the 3D space, but can alternate between several spatial configurations of low-energy called *conformations*. The mainstream approach to conformational analysis is to represent a molecule as a set of structures, called conformers, sampled from its class of admissible conformations. On the methodological side, this operation casts the learning problem into the framework of *multi-instance learning*, that has been drawing a considerable interest in the machine learning community since its initial formulation [53]. The SVM and kernel approaches lend themselves particularly well to this problem, due, on the one hand, to extensions of the SVM algorithm [54], and, on the other hand, to the possibility to define kernels between sets of structures from a kernel between individual structures [55]. A possible solution to the latter approach consists in averaging kernel values over all possible pairs of conformers. While more elaborated schemes can be adopted (*e.g.*, [56]), this simple configuration was already shown to be efficient in practice by Azencott *et al.* [34].

7 - DISCUSSION

As a conclusion, it is probably fair to say that the empirical evaluations of the different kernel constructions introduced in this paper demonstrate the relevance of the approach based on structure kernels for virtual screening. Indeed, on the different tasks they have been tested on, including notably the prediction of mutagenicity [29-32,45], toxicity [29,31,32], anti-cancer activity [31,32] and drug-target inhibition [33,34], these kernels often compare favorably to state-of-the-art approaches. Moreover, because of the intrinsic modularity of kernel methods, this approach offers, to some extent, a unified approach to SAR and virtual screening, for two reasons. First, because they circumvent the need of selecting and extracting molecular descriptors, these kernels can straightforwardly be used to model different biological properties. Second, although we focused in this paper on classification applications, these kernels can be used in conjunction with the whole family of algorithms called kernel methods to solve a great variety of tasks which are relevant for virtual screening and chemoinformatics applications, such as, for instance, regression, clustering and similarity analysis. Concerning its practical use for the screening of large datasets however, it must be stressed that the approach based on kernel methods can be computationally demanding, even for relatively small datasets. Speeding up SVM and kernel methods for large datasets is currently a topic of interest in the machine learning community, and applications in virtual screening on large databases of molecules will certainly benefit from the advances in this field. The choice of a particular kernel, or even more importantly, of the 2D or 3D representation of molecular structures, should be dictated by the application considered. For example, while it is widely accepted that several drug-like properties, such as intestinal absorption [57] or mutagenicity [58] for instance, can be efficiently deduced from the 2D structure of the molecule, target binding prediction is known to depend on a precise 3D complementarity between the structures of the drug and the target, from both the steric and electrostatic perspectives [59]. Nevertheless, even in such problems that intrinsically depend on tridimensional mechanisms, it is not clear that models based on 3D kernels are more efficient than models based on 2D kernels. This fact is especially emphasized in the study of Azencott *et al.* [34] where 2D kernels are shown to outperform 3D kernels (including 3D kernels based on multi-conformers) in general, which actually tallies previous fingerprint-based studies [60,48].

In terms of future work, we see many potential extensions to the general kernel constructions presented above:

- First, the fact that the models could benefit from simple data enrichment schemes, based, for instance, on Morgan indices in the 2D case and partial charges in the 3D case, suggests that the introduction of a more thorough chemical knowledge could improve the expressive power of the kernels. In particular, several reduced representations of molecular structures exist, defined, for instance, by merging aromatic cycles and atoms that are part of the same functional groups in the 2D representation [61], or by considering generic pharmacophoric features instead of isolated atoms in the 3D case [52]. Applying such transformations in a pre-processing step is most likely to improve the characterization of the molecular structures in the kernels, while reducing their computational cost.
- Other important issues that, in our opinion, would be worth studying in more details are related to conformational analysis, and more precisely to the way the conformational space of a molecule is sampled and multi-instance kernels are defined. Although in their current form 3D kernels tend to be outperformed by their 2D counterparts [34], we believe that a proper handling of multi-conformers, together with a higher level of pharmacophoric characterization of molecules, can have a great impact for virtual screening applications.

- Another possible extension would be to adopt a global representation of molecules and to integrate the information derived from their 1D, 2D and 3D structures. A possible approach would be to consider a single kernel defined as a linear combination of kernels for 2D and 3D structures, together with a simple kernel based on global physicochemical properties. Several methods have been proposed to optimize such a kernel combination within the framework of support-vector machines based, for instance, on semi-definite programming [62].
- Finally, in the case of drug-target prediction when additional information about the structure of the target is available, it would be interesting to combine the ligand- and the structure-based approaches to virtual screening, that would most likely benefit to each other in this context.

Last but not least, note that this gentle introduction to kernels for molecular structures and virtual screening applications only reflects our own view and experience, and was deliberately biased towards our own developments in this field. Indeed it must be stressed that, following the pioneer introduction of graph kernels by Kasima *et al.* [27] and Gärtner *et al.* [28], several alternative kernel constructions have been proposed in recent years, among which:

- A graph kernel based on the detection of cyclic- and tree- patterns by Horváth *et al.* [63].
- A graph kernel based on the count of common paths by Borgwardt and Kriegel [64]. However, because it is not possible to consider exhaustive sets of paths, as mentioned in Section 3.2, the kernel construction is restricted to the sets of shortest paths between pairs of vertices.
- An *optimal assignment kernel*, based on the idea of optimally assigning the atoms from one molecule to those of another, by Fröhlich *et al.* [65]. This kernel formulates as the sum of a kernel between pairs of atoms that has to be maximized over all possible assignment of the set of atoms of the smaller molecule to the set of atoms of the bigger one. Unfortunately, albeit very natural, this kernel is not positive definite and might require additional tricks to be used with kernel methods[1].
- Finally, borrowing techniques from computational geometry, standard walk-based graph kernels have recently been extended to kernels between tridimensional structures, based on graphs approximating molecular surfaces by Azencott *et al.* [34].

Together with the references given in the above presentation, this list constitutes, to our knowledge, a comprehensive view of kernel for molecular structures with applications in virtual screening. As an ending remark, we would like to mention that open-source implementations of the family of kernels introduced in this paper can be found within the C++ `ChemCpp` toolbox, freely and publicly available at http://chemcpp.sourceforge.net. We hope that this introductory presentation, together with the availability of this software, will help and motivate the chemoinformatics community to further investigate SVM and molecular kernels to model structure-activity relationship.

[1] Vert, J.-P., The optimal assignment kernel is not positive definite, Technical Report HAL-00218278, **2008**.

ACKNOWLEDGEMENTS

The work related to this paper was done while PM was a PhD student in the Center for Computational Biology of the Ecole des Mines de Paris (France). PM would like to thank XRCE for letting him spend some time writing this paper.

REFERENCES

[1] Hastie, T.; Tibshirani, R.; Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*; Springer, **2001**.
[2] Willett, P. *Drug Discov. Today*, **2006**, *11*, 1046-1053.
[3] Ivanciuc, O. *Anal. Chim. Acta*, **1999**, *384*, 271-284.
[4] Goulon, A.; Picot, T.; Duprat, A.; Dreyfus, G. *SAR QSAR Environ. Res.*, **2007**, *18*, 141-153.
[5] Vapnik, V. N. *Statistical Learning Theory*; Wiley: New-York, **1998**.
[6] Schölkopf, B.; Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press, **2002**.
[7] Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*; Cambridge University Press, **2004**.
[8] Schölkopf, B.; Tsuda, K.; Vert, J.-P. *Kernel Methods in Computational Biology*; MIT Press, **2004**.
[9] Burbidge, R.; Trotter, M.; Buxton, B.; Holden, S. *Comput. Chem*, **2001**, *26*, 4-15.
[10] Weston, J.; Pérez-Cruz, F.; Bousquet, O.; Chapelle, O.; Elisseeff, A.; Schölkopf, B. *Bioinformatics*, **2003**, *19*, 764-771.
[11] Arimoto, R.; Prasad, M.-A.; Gifford, E.M. *J. Biomol. Screen.*, **2005**, *10*, 197-205.
[12] Briem, H.; Günther, J. *ChemBioChem*, **2005**, *6*, 558-566.
[13] Liu, H. X.; Zhang, R. S.; Yao, X., J.; Liu, M. C.; Hu, Z. D.; Fan, B. T. *J. Comput.-Aided Mol. Des.*, **2004**, *18*, 389-399.
[14] Saeh, J.; Lyne, P.; Takasaki, B.; Cosgrove, D. *J. Chem. Inf. Model.*, **2005**, *45*,1122-1133.
[15] Tobita, M.; Nishikawa, T.; Nagashima, R. *Bioorg. Med. Chem. Lett.*, **2005**, *15*, 2886-2890.
[16] Kramer, S.; Frank, E.; Helma, C. *SAR QSAR Environ. Res.*, **2002**, *13*, 509-523.
[17] Helma, C.; Cramer, T.; Kramer, S.; De Raedt, L. *J. Chem. Inf. Comput. Sci.*, **2004**, *44*, 1402-1411.
[18] Luan, F.; Zhang, R.; Zhao, C.; Yao, X.; Liu, M.; Hu, Z.; Fan, B. *Chem. Res. Toxicol.*, **2005**, *18*, 198-203.
[19] Byvatov, E.; Fechner, U.; Sadowski, J.; Schneider, G. *J. Chem. Inf. Comput. Sci.*, **2003**, *43*, 1882-1889.
[20] Müller, K.-R.; Rätsch, G.; Sonnenburg, S.; Mika, S.; Grimm, M.; Heinrich, N. *J. Chem. Inf. Model.*, **2005**, *45*, 249-253.
[21] Takaoka, Y.; Endo, Y.; Yamanobe, S.; Kakinuma, H.; Okubo, T.; Shimazaki, Y.; Ota, T.; Sumiya, S.; Yoshikawa, K. *J. Chem. Inf. Comput. Sci.*, **2003**, *43*, 1269-1275.
[22] Doniger, S.; Hofmann, T.; Yeh, J. *J. Comput. Biol.*, **2002**, *9*, 849-864.
[23] Aires-de Sousa, J.; Gasteiger, J. *J. Comb. Chem.*, **2005**, *7*, 298-301.
[24] Lind, P.; Maltseva, T. *J. Chem. Inf. Comput. Sci.*, **2003**, *43*, 1855-1859.
[25] Liu, H. X.; Zhang, R. S.; Yao, X. J.; Liu, M. C.; Hu, Z. D.; Fan, B. T. *J. Chem. Inf. Comput. Sci.*, **2004**, *44*, 161-167.
[26] Ivanciuc, O. *Reviews in Computational Chemistry*: Wiley-VCH, **2007**, Vol. *23*, Chap. 6, pp. 291-400.
[27] Kashima, H.; Tsuda, K.; Inokuchi, A. In *Proceedings of the Twentieth International Conference on Machine Learning*; AAAI Press, **2003**, pp. 321-328.
[28] Gärtner, T.; Flach, P.; Wrobel, S. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*, Lecture Notes in Computer Science: Heidelberg, **2003**, vol. *2777*, pp.129-143.
[29] Kashima, H.; Tsuda K.; Inokuchi, A. In *Kernel Methods in Computational Biology*; MIT Press, **2004**, Chap.*7*, pp. 155-170.
[30] Mahé, P.; Ueda, N.; Akutsu, T.; Perret, J.-L.; Vert, J.-P. *J. Chem. Inf. Model.*, **2005**, *45*, 939–951.

[31] Ralaivola, L.; Swamidass, S. J.; Saigo, H.; Baldi, P. *Neural Netw.*, **2005**, *18*, 1093-1110.
[32] Swamidass, S. J.; Chen, J.; Bruand, J.; Phung, P.; Ralaivola, L.; Baldi, P. *Bioinformatics*, **2005**, *21*, *S1*, i359-i368.
[33] Mahé, P.; Ralaivola, L.; Stoven, V.; Vert, J.-P. *J. Chem. Inf. Model.*, **2006**, *46*, 2003-2014.
[34] Azencott, C.-A.; Ksikes, A.; Swamidass, S.J.; Chen, J.H.; Ralaivola, L.; Baldi, P. *J. Chem. Inf. Model.*, **2007**, *47*, 965-974.
[35] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*; ACM Press, **1992**, pp. 144-152.
[36] Aronszajn, N. *Trans. Am. Math. Soc.*, **1950**, *68*, 337-404.
[37] Yamanishi, Y.; Bach, F.; Vert, J.-P. *Bioinformatics*, **2007**, *23*, 1211-1216.
[38] Lodhi, H.; Saunders, C.; Shawe-Taylor, J.; Cristianini, N.; Watkins, C. *J. Mach. Learn. Res.*, **2002**, *2*, 419-444.
[39] Collins, M.; Duffy, N. *Adv. Neural. Inform. Proc. Syst.*, MIT Press, **2001**, Vol. *14*, pp. 625-632.
[40] Leach, A. R.; Gillet, V. J. *An introduction to chemoinformatics*; Kluwer Academic Publishers, **2003**.
[41] Gasteiger, J. *Handbook of Chemoinformatics*.; Wiley-VCH, **2003**.
[42] Gärtner, T. *Exponential and Geometric Kernels for Graphs*; NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data, **2002**.
[43] Morgan, H. L. *J. Chem. Doc.*, **1965**, *5*, 107-113.
[44] Ramon, J.; Gärtner, T. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, **2003**, pp. 65-74.
[45] Mahé, P.; Vert, J.-P. *Graph kernels based on tree patterns for molecules*; Technical Report ccsd-00095488, HAL, **2006**.
[46] Leslie, C.; Eskin, E.; Noble, W.S. In *Proceedings of the Pacific Symposium on Biocomputing 2002*; World Scientific, **2002**, pp. 564-575.
[47] Vishwanathan, S. V. N.; Borgwardt, K.; Schraudolph, N. *Adv. Neural. Inform. Proc. Syst.*; MIT Press, **2007**. Vol. *19*.
[48] Brown, R.D.; Martin, Y.C. *J. Chem. Inf. Comput. Sci.*, **1997**, *37*, 1-9.
[49] Matter, H.; Pötter, T. *J. Chem. Inf. Comput. Sci.*, **1999**, *39*, 1211-1225.
[50] McGregor, M. J.; Muskal, S. M. *J. Chem. Inf. Comput. Sci.*, **1999**, *39*, 569-574.
[51] Haussler, D. *Convolution Kernels on Discrete Structures*; Technical Report UCSC-CRL-99-10: UC Santa Cruz, **1999**.
[52] Pickett, S. D.; Mason, J. S.; McLay, I. M. *J. Chem. Inf. Comput. Sci.*, **1996**, *36*, 1214-1223.
[53] Dietterich, T. G.; Lathrop, R. H.; Lozano-Perez, T. *Artif. Intell.*, **1997**, *89*, 31-71.
[54] Andrews, S.; Hofmann, T.; Tsochantaridis, I. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*; American Association for Artificial Intelligence, **2002**, pp. 943-944.
[55] Gärtner, T.; Flach, P. A.; Kowalczyk, A.; Smola, A. J. In *Proceedings of the Nineteenth International Conference on Machine Learning;* Morgan Kaufmann, **2002**, pp. 179-186.
[56] Blaschko, M. B.; Hofmann, T. *NIPS 2006 Workshop on Learning to Compare Examples*; **2006**.
[57] Lipinski, C.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. *Adv. Drug Deliv. Rev.*, **2001**, *46*, 3-26.
[58] King, R.D.; Muggleton, S. H.; Srinivasan, A.; Sternberg, M. J. *Proc. Natl. Acad. Sci. USA*, **1996**, *93*, 438-442.
[59] Böhm, H.-J.; Schneider, G.; Mannhold, R.; Kubinyi, H.; Folkers, G. *Protein-ligand interactions*; Wiley, **2003**.
[60] Brown, R. D.; Martin, Y. C. *J. Chem. Inf. Comput. Sci.*, **1996**, *36*, 572-584.
[61] Gillet, V.; Willett, P.; Bradshaw, J. *J. Chem. Inf. Comput. Sci.*, **2003**, *43*, 338-345.
[62] Lanckriet, G. R. G.; Cristianini, N.; Jordan, M.I.; Noble, W. S. In *Kernel Methods in Computational Biology*; MIT Press, **2004**, Chap. *11*, pp. 231-259.
[63] Horváth, T.; Gärtner, T.; Wrobel, S. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*; ACM Press, **2004**, pp. 158-167.
[64] Borgwardt, K. M.; Kriegel, H.-P. In *Proceedings of the fifth International Conference on Data Mining*; IEEE Computer Society, **2005**, pp. 74-81.
[65] Fröhlich, H.; Wegner, J. K., Sieker, F.; Zell, A. In *Proceedings of the 22nd International Conference on Machine Learning*; ACM Press, **2005**, pp. 225-232.

Received: April 29, 2008 **Revised: June 30, 2008** **Accepted: August 29, 2008**