# Kernel functions for molecular structures and their application to virtual screening with Support Vector Machines

Pierre Mahé

**ECOLE DES MINES**
**DE PARIS**

# T H È S E

pour obtenir le grade de
**Docteur de l'Ecole des Mines de Paris**
Spécialité «Géostatistique»

présentée et soutenue publiquement
par
**Pierre MAHE**

le 7 novembre 2006

**FONCTIONS NOYAUX POUR MOLECULES**
**ET LEUR APPLICATION AU CRIBLAGE VIRTUEL**
**PAR MACHINES A VECTEURS DE SUPPORT**

*Directeur de thèse : Jean-Philippe Vert*

Jury

| | | | |
|---|---|---|---|
| Mme | Florence | D'ALCHE-BUC | Rapporteur |
| MM. | Pierre | BALDI | Rapporteur |
| | Stéphane | CANU | Président |
| | Nicolas | FROLOFF | Examinateur |
| | Stéphane | ROBIN | Examinateur |
| | Jean-Philippe | VERT | Examinateur |

# Remerciements

Voici venu le temps des remerciements. Tant de personnes ont contribué à faire de ces trois années une période si agréable et enrichissante que je vais essayer de faire bref.

Ma reconnaissance va en premier lieu à Jean-Philippe. Travailler avec Jean-Philippe est une expérience vraiment enrichissante dans la mesure où il parvient à diriger efficacement les recherches tout en développant un sens – et un goût – de l'autonomie et de la rigueur scientifique. Je pense avoir appris enormément à son contact, et je lui en suis réellement reconnaissant.

Je remercie ensuite les membres du jury – Florence d'Alché-Buc, Pierre Baldi, Stéphane Canu, Nicolas Froloff et Stéphane Robin – pour s'être interessés à ces travaux et avoir pris le temps de lire en détail mon mansuscript de thèse. Merci en particulier à Pierre Baldi pour avoir fait un si long voyage afin d'être présent lors de la soutenance, et à Stéphane Canu pour sa sympathie et sa considération (sans oublier qu'avec le recul c'est un peu grâce à lui que je me suis engagé dans cette thèse!).

Mes remerciement vont ensuite à l'ensemble des centres de bioinformatique et de géostatistique de Fontainebleau. Merci en particulier à Véronique pour avoir su répondre à mes questions – parfois naives – à propos de chimie et de molécules, à François pour les multiples dépannages techniques et à Isabelle et Nathalie pour leur gentillesse et leur efficacité dans le support administratif. Je n'oublie pas tous les thésards, et en particulier Marco, Martial, Franck, Laurent et Misha pour la bio et Caroline, Marco, Mathieu et Marie pour la géostat'...ainsi que les anciens et ceux des centres voisins (Christophe, Alex, Nico, François, Romain, Thimothée, Thibault, Aurélien)...et bien sûr tous ceux que j'oublie! J'ai une pensée particulière pour tous ceux qui ont eu la chance de partager mon bureau...et par là même tous les conflits que j'ai pu avoir avec mon ordinateur.

Durant ces trois années, j'ai eu la chance de travailler avec le centre de bioinformatique de l'université de Kyoto (au passage, merci encore une fois à Jean-Philippe

pour m'avoir impliqué dans ce projet), et de me rendre plusieurs fois au Japon. Je tiens à remercier toutes les personnes que j'ai pu y rencontrer pour leur acceuil et leur gentillesse. Je pense en particulier au professeur Akutsu, à Nobuhisa, Yoshi, Hiroto, Dukka, Koichiro, J.B., José, Lidio...et, une fois de plus, à tous ceux que j'oublie! Une pensée particulière pour Yoshi et Hiroto, avec qui j'ai vraiment passé de bons moments, aussi bien au Japon qu'en France. Je n'oublie pas Jean-Luc qui m'a accueilli chez lui, fait découvrir Kyoto et la gastronomie japonaise...mais également le goût du développement logiciel au travers du projet "ChemCpp" (sans aucun doute LE standard chemoinformatique de demain).

Mais tout cela ne serait rien sans le support constant de ma famille: un grand merci à mes parents, mes grands-parents et ma soeur pour m'avoir toujours supporté dans l'orientation de mes études (je vous avais bien dit que ça s'arrêterait un jour!). Sans oublier bien sûr tous mes amis avec qui j'ai partagé bon nombre de moments agréables au cours de ces trois dernières années. Les lister tous serait bien trop long, aussi soyez sûrs, très chers amis, que je n'oublie personne. Une pensée particulière va néanmoins à mes compagnons successifs de colocation (Charlotte, Nico, Hélène, Vinze & Céline, Helmut), aux normands, aux INSA, aux micro-potes, à Georges Abidbol et ses ouiches lorraines, et bien sûr – "last but not least" comme on dit – à Caroline qui a su me soutenir, m'encourager, me donner confiance en moi...et bien plus encore...et ce malgré mon sale caractère de cauchois!

# Résumé

La recherche thérapeutique a de plus en plus recours à des techniques de modélisation, dites de criblage virtuel, visant à corréler la structure d'une molécule avec ses propriétés biologiques. En particulier, l'utilisation de modèles prédictifs quantifiant la toxicité d'une molécule ou son activité vis à vis d'une cible thérapeutique, permet de réduire de manière considérable le temps et les coûts nécessaires à la mise au point de nouveaux médicaments.

Nous nous proposons d'aborder ce problème dans le cadre des méthodes à noyaux, qui permettent de construire de tels modèles de manière efficace dès lors que l'on dispose d'une fonction noyau mesurant la similarité des objets que l'on considère. Plus particulièrement, l'objet de cette thèse est de définir de telles fonctions noyaux entre structures bi- et tri-dimensionnelles de molécules, ce qui se traduit d'un point de vue méthodologique comme le problème de comparer des graphes représentant les liaisons covalentes des molécules, ou des ensembles d'atomes dans l'espace.

Plusieurs approches sont envisagées sur la base de l'extraction et de la comparaison de divers motifs structuraux permettant d'encoder les groupes fonctionnels des molécules à diférents niveaux de résolution. Les validations expérimentales suggèrent que cette méthodologie est une alternative prometteuse aux approches classiques en criblage virtuel.

# Abstract

Computational models play a role of increasing importance in therapeutic research. For example, accurate predictive models accounting for the biological activity and drug-likeliness of molecular compounds can lead to substantial savings in terms of time and costs for the development of new drugs if they are applied early during the drug discovery process. While processing chemical data in this context involves many different tasks, including for instance clustering, regression, classification or ranking, most of them are related to *Structure-Activity Relationship (SAR) analysis*. In other words, the central problem is to find a relationship between the structures of molecules and their biological activity, under the basic assumption that molecules having a similar structure are likely to exhibit a similar biological behavior.

Decades of research in machine learning and statistics have provided a profusion of models for that purpose. Each of them has its own specificities, and the choice of a particular model has to be related to the final goal of the analysis, usually balanced between criteria of efficiency and interpretability. Nevertheless, a common issue to all models concerns the data representation, or in other words, how to handle molecular structures. Indeed, most machine learning algorithms are vector based, and typically require to encapsulate molecules into vectors of limited size. Prior to building a model, the modeller must therefore resort to chemical expertise or heuristic feature selection methods in order to choose, among the plethora of molecular descriptors, which descriptors are relevant with respect to the property to be predicted.

An alternative direction has been explored recently using the theory of kernel methods. Kernel methods, and in particular Support Vector Machines, are gaining increasing popularity in the machine learning community for their well-sounded formulation and good performance on many real-world problems. This family of algorithms possesses two important properties. First, it theoretically allows learning in very high – potentially infinite – dimensions thanks to a heavy use of regularization ; second, instead of an explicit representation of data as vectors, it only requires inner products between such representations, through what is usually referred to as a positive definite kernel function. Kernel functions stand at the interface between the data and the learning algorithm, and constitute the key element of kernel methods. Because

many real-world data do not have a natural vectorial representation, the problem of defining kernel functions for structured data, such as strings, trees and graphs, has been drawing a considerable attention in the machine learning community in the last few years.

Ushering in this avenue, we consider in this thesis the problem of designing kernel functions for molecular structures. Together with the panoply of kernel methods, this might offer a unified approach to structure-activity relationship analysis, without the need of explicitly extracting molecular descriptors. On the methodological side, this approach usually boils down to defining a measure of similarity between graphs representing molecules as sets of atoms connected by covalent bonds. On the practical side, however, the nature of the graph structure often renders intractable the traditional substructure-based approach to the design of kernel between structured data, that has been, for instance, successfully applied to strings and trees. Nevertheless, several graph kernels that circumvent this complexity issue have recently been proposed. The approach adopted is often that of restricting the degree of complexity of the substructures characterizing the graphs, and, in particular, pioneer graph kernels propose to represent a molecule by a set of simple chains of atoms. In the first part of this thesis, we revisit this pioneer approach with the double goal to reduce its computational complexity and improve its expressive power. This work is cast into a more general framework in the second part of the thesis, where, based on a recently introduced formulation, the set of substructures characterizing the graphs is extended to the class of trees. Finally, motivated by the fact that the tridimensional information plays a central role in many biological mechanisms, we apply, in the third part of the thesis, this general methodology to handle three-dimensional structures of molecules.

# Contents

Chapter 1

# Context

In this preliminary chapter, we lay the general context of this thesis that motivates the design of kernel function between molecular structures. We start by a brief introduction to the field of therapeutic research in Section 1.1, the main goal being to highlight the impact of computational models in its rationalization, and their potential to dramatically decrease the time and cost required to bring a new drug to the market. After discussing several open issues raised by the introduction of computer science in chemistry in Section 1.2, we turn to the main topic of the thesis in Section 1.3 with the presentation of the Support Vector Machine (SVM) algorithm. SVM and kernel methods have been drawing considerable attention in the machine learning community over the last decade because of their good performance on many real world applications. Chemical informatics does not disrespect this trend, and Section 1.4 illustrates the increasing popularity gained by SVM for the prediction of biological properties of molecules. A particular emphasis is put in this latter section on the design of kernel functions for molecular structures, that offers an alternative to their traditional vector-based representation and constitutes the central topic of this thesis. Finally, Section 1.5 summarizes our contributions for that purpose, and draws the outline of the thesis.

## 1.1 Introduction to modern therapeutic research

This section is meant to give a quick overview of therapeutic research to nonspecialists. After a general introduction to the drug discovery process in Section 1.1.1, we focus on its evolution over the last two decades. In the 1980's, the development of high-throughput technologies that we introduce in Section 1.1.2 was expected to solve the drug discovery problem by a massive parallelization of the process. In practice, it turned out that, if they were not carefully deployed, these new technologies could lead to such a tremendous increase of candidate molecules that the drug discovery process became like finding a needle in a haystack. As a result, the large-scale approach has been progressively abandoned over the recent years, for the profit of more rationalized

process, in which the computational methods introduced in Section 1.1.3 have gained a role of increasing importance.

### 1.1.1 Basics of drug discovery

Developing a therapy for a particular disease is a process usually articulated around three steps. The first step of target identification consists in the identification of a biological molecule, often a protein, involved in a mechanism leading to the disease. The purpose of the second step is to identify a small molecule with an interesting biological profile that is able to interfere with this therapeutic target. Eventually, before this drug candidate enters the market, the third step of clinical validation must demonstrate its efficiency and safety through an extensive evaluation on animals and humans.

#### Biological target

The primary goal in therapeutic research is to interfere with a metabolic or signaling pathway responsible for a disease. Metabolic and signaling pathways are cascades of chemical reactions occurring within the cell, that lead respectively to the formation of a metabolic product to be used by the cell, or to the alteration of gene expression by the activation of transcription factors. The task of therapeutic research is to find a drug molecule able to modify such a pathway by the alteration of a key entity, usually a protein, involved in the corresponding cascade of reactions: the *therapeutic target*. Target identification involves both biological and chemical knowledge, in order to discover potential targets and assess their "drugability", that is, to what extent they can be altered by a drug molecule (Drews, 2000).

While current drug therapies address only about 500 biological targets, 45% of which belong to the class of G-protein coupled receptors (GPCR), a family of cell membrane proteins, and 28% are enzymatic proteins, it is assumed that at least 10 times more targets could be exploited for future therapies (Drews, 2000). The process of target identification is likely to benefit from the recent completion of the Human Genome Project and the advent of high throughput gene activity monitoring devices. In particular, potential therapeutic targets may directly be identified from the difference of proteins expression observed in microarray experiments involving healthy and diseased populations, and the number of identified targets is expected to dramatically increase over the next few years (Debouck and Goodfellow, 1999; Chanda and Caldwell, 2003).

Before turning to the step of drug discovery itself, the identified target must be *validated* in order to demonstrate its critical role in the disease. Target validation usually involves in-vitro and/or in-vivo experiments, typically based on gene knocking-out experiments and transgenic animal models.

### Drug discovery

In a second step, the goal is to find a small molecule, called a *ligand*, able to bind by intermolecular forces to the target and alter its normal functioning. This interaction is said to be *direct* when the drug binds to the active site of the target and competes with its natural substrate, or *indirect* if the drug binds to a secondary site and induces changes in the chemical conformation of the target, thereby modulating its affinity with its natural ligand. For example, these two strategies have been exploited by drugs targeting HIV reverse transcriptase, either targeting the active site of this enzyme, or a secondary site (Ren and Stammers, 2005).

In order to quantify the *activity* of the ligand, corresponding to its degree of interaction with the target, an experimental setup known as a *biological assay for activity* must be designed. In the case of an enzymatic target for instance, the activity of the ligand corresponds to its degree of competition with the natural substrate of the enzyme, usually expressed as the concentration at which it reduces by 50% the catalytic action of the target, a quantity known as IC50.

Molecular compounds can subsequently be *screened* for activity in order to find promising drug candidates, or *lead compounds*, able to interfere with the target at low concentration rates. The number of candidate molecules with simple drug characteristics that could be synthesized is theoretically assumed to be of the order of $10^{60}$ (O'Driscoll, 2004). The identification of promising candidates among this vast (almost infinite) amount of molecules strongly relies on biochemical expertise and is traditionally achieved in an iterative process, sometimes denoted as the *drug discovery cycle*, alternating between steps of selection, possibly synthesis, and screening of candidates, the results of the screening guiding in turn the next selection step (Manly et al., 2001). Early screens of the drug discovery cycle identify *hits*: molecules showing chemical activity, but not necessarily fulfilling the efficiency requirement of leads (Jorgensen, 2004). To this *hit generation* follows a *hits to leads* phase, where the identified hits are validated by confirmation screens, and possibly structurally refined in order to increase their potency with respect to the target. If sufficient potency is attained, additional counter screens may be applied to ensure that the lead candidates do not interact with homologous proteins of the target, with the goal to limit their side effects.

At this point, lead compounds with good binding abilities are identified. However, binding to the target is not a sufficient condition for the identification of a promising drug candidate. Indeed, not only must a drug interfere with the therapeutic target, but it must also have a good biological profile, and in particular a low toxicity in order to be harmless to the organism, and good pharmacokinetics properties. Broadly speaking, pharmacokinetics is related to the behavior of the drug in the body, such as its ability to enter the bloodstream and reach the target, and to be further destroyed and eliminated by the body. Major pharmacokinetics properties are compiled in the acronym *ADME*, standing for Absorption, Distribution, Metabolism and Excretion (Xu and Hagler, 2002; Boobis et al., 2002). The final step of drug discovery is a phase of *lead optimization*, where the chemical structure of the leads is refined in order to meet

these *drug-like* criteria. Contrary to screening assays, usually carried out in vitro, lead optimization often involves in vivo experiments on animals. This optimization process is highly iterative and is considered to be a major cause of failures of the drug discovery process. When a lead compound with promising drug-likeliness is discovered, the final step toward a marketed drug is a phase of clinical validation.

**Clinical validation**

Before entering the market, the drug candidate must be validated by an extensive testing phase, aiming to demonstrate its efficiency and safety for the human organism: the *clinical validation*. Clinical validation starts with preliminary tests for safety on animals, the *pre-clinical* step, and is subsequently articulated around three phases (DiMasi et al., 2003):

- Phase I (1-2 years) : tests for safety are first carried out with a limited number ($< 100$) of healthy persons.
- Phase II (1-2 years): tests for safety and efficiency are then applied to a larger pool of several hundred persons, from both the healthy and diseased groups.
- Phase III (2-3 years): finally, a large scale efficiency test, involving a larger pool of persons (in the thousands) from different demographic areas, completes the study.

Eventually, provided this clinical study obtains necessary government approvals, delivered for example by the Food and Drug Administration (FDA) in the USA and the European Agency for the Evaluation of Medical Products (EMEA) in Europe, commercial exploitation of the drug molecule can begin.

As a conclusion, it is worth stressing that therapeutic research is a very complex process, that raises tremendous costs in terms of time and money. A recent study involving 68 drugs approved in the 1990's in the USA, showed that the average time and cost of drug development, from the step of target identification to the obtaining of FDA approvals, were nearly 15 years and US$ 900 millions (DiMasi et al., 2003). Over the last two decades, however, technological breakthroughs, together with a progressive reconsideration of the process itself, has led to major revolutions in the process of finding a drug.

### 1.1.2 High-throughput technologies

From now on, we focus on the problem of drug discovery itself, that is, the identification of promising clinical candidates for a validated therapeutic target. Until the 1980's, due to the cost (both in time and money) of synthesizing, and testing new molecules, the step of "hits generation" was the main bottleneck of the drug discovery

process (Bleicher et al., 2003; Xu and Hagler, 2002). Advances in miniaturization and robotization in the 1980's led to the appearance of *high-throughput screening (HTS)* devices, allowing to massively screen in parallel pools of hundreds or thousands of molecules, and nowadays, ultra-HTS allows major pharmaceutical company to screen up to 100 000 molecules a day (Armstrong, 1999; Bajorath, 2002). In response to this new technology, chemical research developed at that time methods to synthesize molecules in parallel, thereby offering the possibility to generate large libraries of molecules. In particular, *combinatorial chemistry* makes it possible to create a large number of molecules from the combination of elementary building blocks at relatively low cost (Lazo and Wipf, 2000; Miertus et al., 2000).

The synergy between these two technologies was expected to facilitate the drug discovery cycle by a systematic screening of large pools of compounds (O'Driscoll, 2004; Xu and Hagler, 2002). In practice however, it was soon realized that this large scale approach was not an answer to the drug discovery problem. While the number of identified hits could substantially be increased, no corresponding growth in the number of drugs entering the market was observed, and people realized that the real bottleneck of the drug discovery process lied in the steps of lead identification and optimization (Drews, 2000; Bleicher et al., 2003). This observation progressively led to a reconsideration and rationalization of the drug discovery process, in which computational models have gained a role of tremendous importance (Manly et al., 2001).

### 1.1.3 Computational models

With the necessity to exploit the massive amount of data generated by high-throughput technologies, computer science methods are being increasingly used in the drug discovery process and in chemistry in general. In order to unify the blend of computer science methods and chemistry, F.K Brown coined in 1998 the term *chemoinformatics* as

> "Chemoinformatics is the mixing of those information resources to transform data into information and information into knowledge for the intended purpose of making better decisions faster in the area of drug lead identification and optimization."

This general definition encompasses many aspects, and in particular the representation, storage, retrieval and analysis of chemical information. Most chemoinformatics techniques are well established from decades of academic and industrial research, and it has been suggested that chemoinformatics is just "a new name for an old problem" (Hann and Green, 1999). The field of application of chemoinformatics methods in therapeutic research is very large. In this section, we introduce three major steps taken toward the rationalization of the drug discovery process. We refer to the textbooks (Leach and Gillet, 2003; Gasteiger and Engel, 2003) for a wider overview of chemoinformatics methods.

### Library design

It was soon realized that the number of compounds that could be generated by combinatorial chemistry was so huge that systematically screening random combinatorial libraries was unlikely to discover lead compounds. This observation motivated a rationalization of the screening process in order to limit the number of HTS assays, with the view to reduce the time and costs of the step of lead identification. The focus has now changed from the screening of large combinatorial libraries to a process of *library design* in order to define smaller, but information rich libraries (Hann and Green, 1999; Bajorath, 2002; Manly et al., 2001). Because the goal of the screening evolves during the drug discovery process, the design of the libraries to be screened must be related to the advancement of the project. In order to maximize the probability of identifying structurally different hits, early screening assays must involve *diverse libraries* giving a broad coverage of the chemical space (Xu and Hagler, 2002). On the contrary, in the later "hits to lead" step, *targeted libraries* made of molecule structurally similar to the identified hits must be designed (Manly et al., 2001).

Library design is therefore concerned with the spread in the chemical space of the libraries to be screened. Standard algorithms involved in library design are based on a partitioning of the chemical space, or on a clustering of molecules in sets of similar molecules (Hann and Green, 1999; Bajorath, 2002). Diverse libraries can for instance be defined by picking representative elements from the bins of the chemical space partition, or the cluster of molecules, and targeted libraries can directly be obtained from the bins, or the clusters, the hits belong to. On the methodological side, this type of algorithms is based on a measure of similarity between molecules in order to define neighborhood relationships in the chemical space. Assessing molecular similarity has been drawing considerable attention in chemoinformatics, and is the topic of Section 1.2.2.

### Virtual screening

*Virtual screening*, or *in silico* screening, is the term broadly used to denote the computational analysis of databases of compounds aiming to identify candidates having the desired activity for a specific therapeutic target. This can be seen as an alternative to performing wet-lab experiments with the key advantages that arbitrary large amounts of molecules can be considered, molecules need not to be synthesized, and even *virtual* molecules, that is, molecules that cannot be synthesized yet, can be evaluated. With the identification of potentially active compounds, virtual screening can therefore help reduce the number of screening assays, and motivate the purchase or synthesis of new molecules (Manly et al., 2001; Bajorath, 2002). On the practical side, virtual screening methods require either the knowledge of the structure of the therapeutic target, usually obtained by crystallography, or the measured activity of a set of compounds.

If the structure of the target is known, the most common approach to virtual screen-

ing is *docking*, which consists in deriving an activity score from the optimal positioning of the ligand in the active site of the target (Kitchen et al., 2004; Jorgensen, 2004). Docking algorithms consist in two core components: a *search algorithm*, in charge of positioning the ligand, and an *evaluation function*, quantifying the strength of the resulting binding in terms of energy, through the evaluation of intermolecular forces. The search algorithm needs to optimize the three dimensional configuration of the ligand, its *conformation*, with respect to the spatial and electrostatic structures of the active site. The docking approach to virtual screening consists in docking all the molecules from the database, and selecting the top scored molecules as the most promising active molecules. Because of the presence of rotational bonds, molecules usually take a large number of conformations, and docking can be a quite complex process in practice. Different strategies can be adopted to explore the *conformational space* of the ligand. The mainstream approach is to consider the ligand as a flexible structure. In *flexible docking* for instance, the optimal conformation is obtained by progressively tweaking the rotational bonds of the molecule. Alternatively, in *incremental docking*, the ligand is first be decomposed into substructures, and is subsequently incrementally grown in the active site from the docking of its substructures. The opposite approach considers a set of stable conformations of the ligand, and lets the score to the ligand be the highest score obtained by its conformations in a *rigid docking*, based on global operations of translation and rotation only. While they are considered to be less predictive than their flexible counterparts, rigid methods are computationally cheaper and easier to automate, and for this reason, they can very useful to rapidly remove molecules of low activity in a preliminary docking step (Kellenberger et al., 2004). Common docking software are GOLD[1] for the flexible docking (Jones et al., 1997), FlexX[2] for the incremental docking (Rarey et al., 1996) and FRED[3] for the rigid docking of multi conformers (Miteva et al., 2005).

Alternatively, *de novo design* can be used to find potential inhibitors from scratch given the structure of the target. De novo algorithms generate a series of candidate molecules obtained by the addition of substituents to a predefined core structure fitting in the binding site. Substituents are chosen from a set of typical drug fragments, and their addition to the core structure must lead to a synthetically available molecule having a good complementarity with the binding site of the target (Kitchen et al., 2004; Jorgensen, 2004).

If the structure of the target is unknown, virtual screening methods can be derived from a pool of compounds of known activity obtained by preliminary screenings experiments. This is known as the *ligand-based* approach to virtual screening, in opposition to the above *structure-based* approach. A simple ligand-based approach consists in ranking the molecules from the dataset with respect to their similarity to the set

[1] `http://www.ccdc.cam.ac.uk/products/life_sciences/gold/`

[2] `http://www.biosolveit.de/FlexX/`

[3] `http://www.eyesopen.com/products/applications/fred.html`

of known active compounds, and choosing the most promising candidates as the top ranking ones (Jorgensen, 2004). This approach is similar in essence to the clustering and partitioning algorithms involved in the problem of targeted library of the previous section. However, the approach is more general here since the simple criterion of proximity in the chemical space can be replaced by more evolved ranking schemes taking simultaneously into account the whole set of active compounds (Wilton et al., 2003).

Alternatively, the set of active compounds can be used to derive a *pharmacophore model* representing a three dimensional arrangement of molecular features responsible for activity (or at least thought to be). This pharmacophore model can then be used as a *filter* to remove from the dataset the compounds not fulfilling this necessary condition of activity (Jorgensen, 2004; Finn et al., 1998).

Because their focus is on the active compounds only, the above approaches can be criticized by the fact that they do not take fully advantage of the screening experiments (Manly et al., 2001). The more general approach consists in building a model correlating the structure of the molecules with their biological activity from the whole pool of screened molecules, thereby integrating information about the activity as well as the inactivity of the compounds, which is ignored in the above approaches. This problem is known as modeling *Structure-Activity Relationship (SAR)*, and involves methods from the fields of statistics and machine learning. We postpone its introduction to Section 1.2.3.

When the structure of the target is known, the structure-based approach naturally stands as the more rational way to virtual screening, and was successfully deployed in several recent drug development programs (see Klebe, 2000, and references therein). It must be stressed that the ligand-based approach is nevertheless of considerable interest for at least two reasons. First, the docking approach is computationally heavy and hard to automate, which limits in practice the size of the databases of molecules that can be considered (Bleicher et al., 2003). Second, an even more striking point is the fact that, in many cases, the structure of the target is difficult, if not impossible, to obtain, and the docking approach is not even applicable. In particular, this is the case for the class of GPCR targets, that constitute almost 50% of the current therapeutic targets, and this situation is actually likely to become the common case in the near future, since the number of potential therapeutic targets is expected to dramatically increase with recent advances in the fields of genomics and bioinformatics (Manly et al., 2001).

As a conclusion, we can note the strong synergy between virtual and physical screenings: screening assays supply valid data to build, or refine, the virtual screening models, that in turn optimize the screening assays. Combining virtual and high-throughput screening in an iterative manner leads to a process of *sequential screening* (Bajorath, 2002; Manly et al., 2001; Warmuth et al., 2003). While this synergy is particularly obvious in the ligand-based approach, this sequential scheme may as well be helpful to refine a docking model, where the optimization of the evaluation function

is known to be a difficult task in practice (Kitchen et al., 2004).

### Early integration of drug-likeliness

It is now widely accepted that the major cause of failure in drug discovery is the poor drug-likeliness, that is, the bad pharmacokinetics and toxicity properties, of the clinical candidates (Bleicher et al., 2003; Xu and Hagler, 2002; Butina et al., 2002; Manly et al., 2001). This observation revealed that in the phase of lead optimization, the ability to improve the drug-likeliness of candidates selected on the sole basis of their activity was overestimated, and led people to integrate the notion of drug-likeliness earlier in the process of lead identification (Manly et al., 2001; Bleicher et al., 2003). This reconsideration of the sequential paradigm of lead identification and optimization marks a major advance toward the rationalization of the drug-discovery process, which is likely to severely reduce clinical attrition, and, as a consequence, the overall cost the process (Bleicher et al., 2003; DiMasi et al., 2003).

The seminal example of early drug-likeliness integration is due to Lipinski and co-workers who proposed a simple set of rules, known as the *rules of five*, relating the propensity of a molecule to be absorbed by the organism to general molecular properties, such as the molecular weight and the number of hydrogen bonds donors and acceptors (Lipinski et al., 2001). These rules are commonly used as a *filter* to remove compounds from the drug discovery pipeline, molecules being unlikely to be absorbed if they violate at least two rules. Similar filters were proposed to address other drug-like properties, in particular in the context of toxicity, where potentially toxic molecules can be identified by the detection of *toxophores*, that is, reaction groups responsible for toxicity (Bajorath, 2002; Manly et al., 2001).

Following the SAR approach involved in virtual screening, statistical and machine learning models that correlate directly the structure of the molecules with their drug-like properties are also useful in this context (Butina et al., 2002; Boobis et al., 2002). Strictly speaking, these models are not related to SAR analysis since the term activity refers to the notion of binding to the target, and the terminology usually adopted is that of *Structure-Property Relationship (SPR)* analysis. Nevertheless, integrating the notion of drug-likeliness in the process of lead identification can be seen as casting the problem of virtual screening and SAR analysis, into a more general *multi-dimensional*, or *multi-property* optimization framework (Ekins et al., 2002; Bleicher et al., 2003; Xu and Hagler, 2002), and the terms virtual screening and SAR often reflect this broader meaning.

In conclusion, it must be clear that these methods are highly complementary in the process of identifying lead candidates with promising drug-likeliness. In the case where the structure of the target is unknown, Figure 1.1 gives a schematic view of their possible integration in the drug-discovery process.

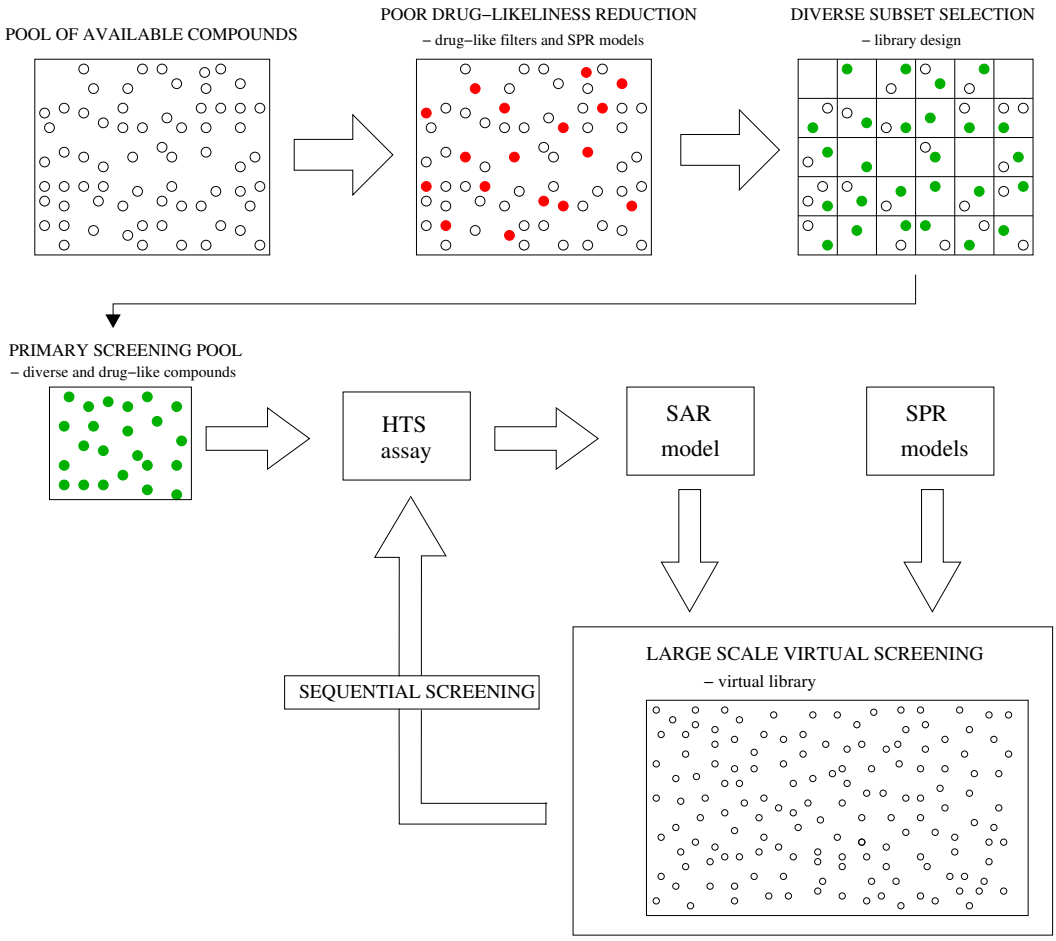


Figure 1.1: A possible integration of computational methods in the drug discovery process. Molecules with poor drug-like properties are first removed from a pool of available compounds using SPR models and/or drug-like filters. A primary screening pool is subsequently defined by choosing a diverse subset of candidates. The results of the screening help building a SAR model, which, in conjunction with SPR models, can be used in a sequential screening approach to discover lead compounds with promising drug-likeliness.

## 1.2 Open issues in chemoinformatics

The introduction of computer science in chemistry raises a lot of practical issues. In this section we focus on the problems of representing molecular structures, assessing their similarity and the prediction of molecular properties, that are central to virtual screening applications.

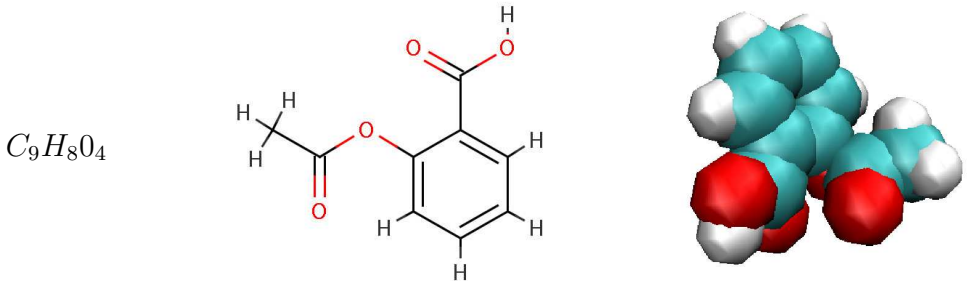


Figure 1.2: 1D, 2D and 3D representations of the aspirin molecule.

### 1.2.1 Representation of molecular structures

Different perspectives can be adopted to represent a molecular structure:

- the simplest 1D representation is defined from the atomic constitution of the molecule.

- the 2D representation is a graph called *molecular graph*, representing the covalent bonds between the atoms of the molecule.

- finally, in the 3D space, a molecule can be seen as a *skeleton*, resulting from the introduction of atomic coordinates in the molecular graph, or as various *molecular surfaces* representing the projection of physicochemical properties of the atoms, such as their electrostatic potential, hydrophobicity, charge or Van der Waals volume, on a sphere enclosing the molecule (Gasteiger and Engel, 2003, section 2.10). Theses surfaces are in particular of great interest in docking applications in order to quantify the complementarity between a ligand and a target. However, molecules are not inert elements, and because of the presence of rotational bonds, they can take several low-energy spatial configurations known as *conformations*. Considering the 3D structure of a molecule therefore raises the problem of *conformational analysis*, which boils down in practice to representing the molecule as a set of conformations, in an approach called *multi-conformers*, or as a flexible structure, in which case a structural degree of freedom is introduced for each rotational bond.

Figure 1.2 presents different representations of the aspirin molecule. On the practical side, chemoinformatics methods are often unable to handle directly these representations, and require to encapsulate the structural information into a set of *descriptors*. Molecular descriptors can be classified into three categories, depending on the dimensionality of the representation from which they are derived.

### 1D / constitutional descriptors

1D descriptors are derived from the atomic composition of the molecules. They correspond typically to global molecular properties, such as the molecular weight and hydrophobicity, or general constitutional features, such as the number of atoms of particular types or hydrogen bond donors and acceptors. These descriptors bear little information about the structure of the molecule and are essentially used to derive filters such as the Lipinski's rule of five, or in combination with other descriptors.

### 2D / topological descriptors

2D descriptors are derived from molecular graphs. A first class of 2D descriptors consists of general *topological indices*, related to the notion of graph invariant of graph theory. Seminal examples include the Wiener and Randic connectivity indices, defined respectively from the length of the shortest path between pairs of atoms, and their degrees in the molecular graph (Gasteiger and Engel, 2003). In a related approach, *topological autocorrelation vectors* measure the autocorrelation of atomic physicochemical properties, such as partial charges or polarity, from pairs of atoms separated by a given topological distance, expressed as the length of the shortest path connecting atoms in the molecular graph (Moreau and Broto, 1980).

A second class of descriptor represents a molecule by a vector indexed by a set of structural features, and relies on the extraction of substructures from the molecular graph. This process defines a *molecular fingerprint*, and in practice, two different approaches can be adopted. The first approach considers a limited set of informative structures called *structural keys* to characterize the molecule. Each structural key is mapped to a bin of the fingerprint, which either accounts for the presence or the frequency of the structure in the molecule. A typical implementation is a bistring indexed by 166 predefined structures known as the *MDL MACCS keys* (McGregor and Pallai, 1997). In the alternative approach, molecules are represented by simple structural features called *linear molecular fragments*, defined as successions of covalently bonded atoms. In this case, typical fingerprints, such as the *Daylight* fingerprints for instance, characterize a molecule by its exhaustive list of fragments made of up to seven or eight atoms. Because the number of fragments occurring in a molecule is typically very large, fragments cannot be mapped to distinct bins of the vector and in practice, hashing algorithms are used to limit the size of the vector, typically to 1000 or 2000 elements. The hashing process maps each fragment to several bins of the vector, thereby inducing a phenomenon of collisions, or "clashes", in the fingerprint. These two representations are very popular in chemoinformatics and are illustrated in Figures 1.3 and 1.4. Among the advantages offered by the structural keys over the hashed fingerprint representation is the expressiveness of the features, the interpretability retained in the representation, because of the one-to-one correspondence between the bins of the vector and the structural features, and the possibility to consider the frequency of the features, which is made impossible in hashed fingerprints by the use of hashing algo-

rithms. On the other hand, hashed fingerprints consider a much larger set of features to represent the molecules and are easier to implement because linear fragments are particularly simple to detect. Moreover, whereas the fingerprint representation is universal, choosing the features to be included in the structural keys representation may be challenging in practice. While chemical intuition can be helpful for that purpose (Xue et al., 2001b), this task is more generally related to the problem of *graph mining* that consists in the automatic identification of interesting structural features within a set of graphs. For chemical applications, such interesting patterns are typically defined as non correlated structures frequently appearing in active compounds, and rarely in inactive compounds (Deshpande et al., 2005; Helma et al., 2004; Inokuchi et al., 2003).

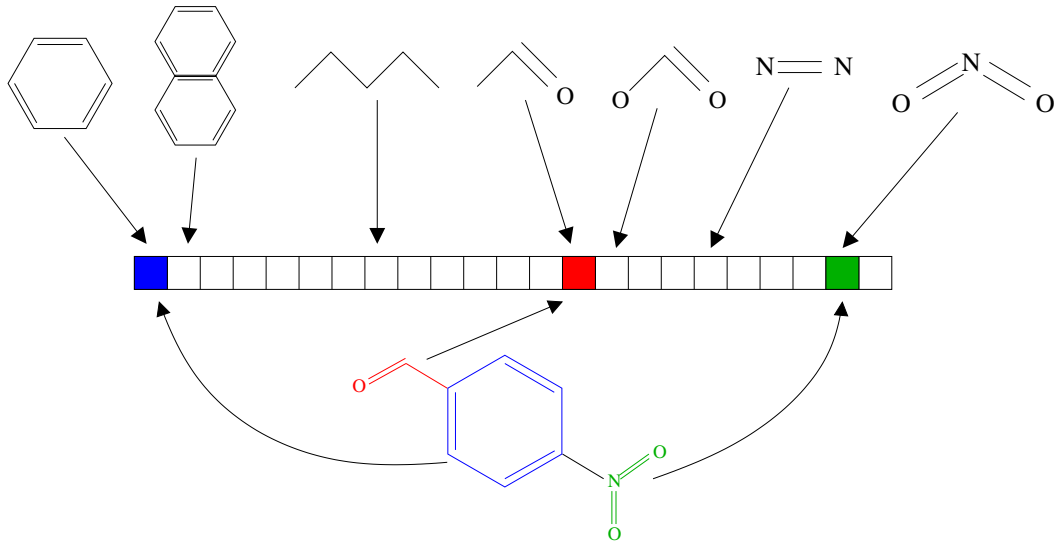


Figure 1.3: Illustration of the structural keys representation. The molecule is represented by a vector indexed by different structural keys

### 3D / geometrical descriptors

3D descriptors are derived from the 3D representation(s) of the molecules. The first class of three dimensional descriptors requires a preliminary step of *molecular alignment*, consisting in placing the molecules in a common orientation in the 3D space through operations of rotations and translations. The quality of the alignment is quantified by a scoring function, and the molecules are said to be aligned when it is maximized. Typical score functions consider the number of identical atoms superimposed under a skeleton representation (Thimm et al., 2004), or the overlap of the electron clouds surrounding the molecules (Bultinck et al., 2003). In order to handle conformational analysis, the alignment can be flexible, in which case additional degrees of freedom are introduced to handle rotational bonds, or rigid and based on the optimal alignment of pairs of multi-conformers. Aligning molecules can be a quite

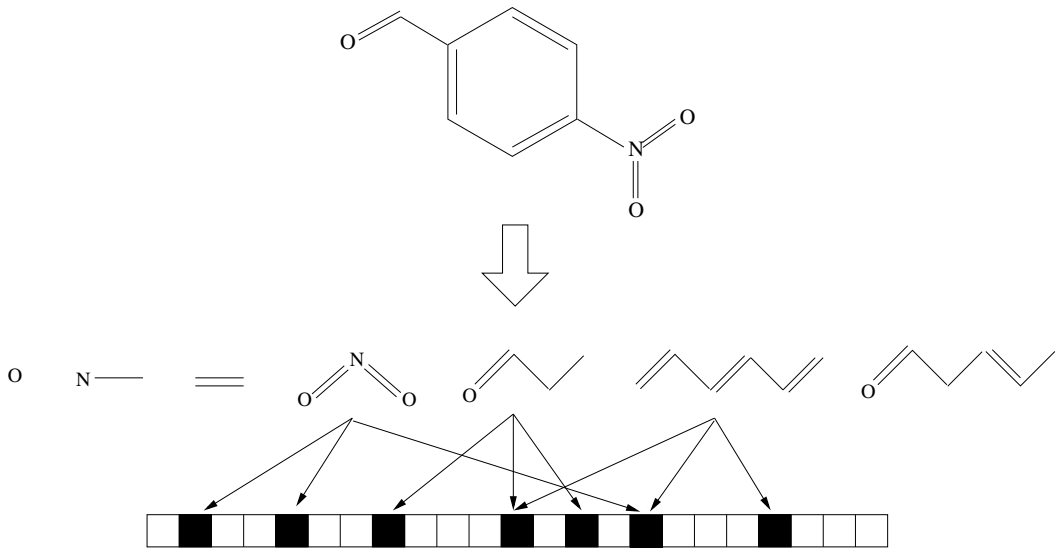

Figure 1.4: Illustration of the hashed fingerprint representation. The molecule is represented by a large set of linear fragments (just a few are represented here), each of them indexing several bits of the vector (3 here), thereby introducing a phenomenon of collision.

complex process, and we refer to Lemmen and Lengauer (2000) for a review of the existing techniques. Once the molecules are aligned, 3D descriptors can for instance be defined by sampling molecular surfaces according to rays emanating from the center of mass of the aligned molecules (Dietterich et al., 1997; Perry and van Geerestein, 1992), or, in the *Comparative Molecular Field Analysis (CoMFA)* methodology, by measuring the interaction between the molecules and an atomic probe (e.g., a charged or lipophilic atom) at each point of a discrete box enclosing the molecules (Kubinyi, 2003).

An opposite approach consists in extracting descriptors independent of the molecular orientation. Apart from global shape descriptors, such as the Van der Waals volume of the molecule or molecular surfaces areas, most alignment independent descriptors are based on distances between atoms. For example, an early study proposed to characterize a molecule by its matrix of inter-atomic distances (Pepperrell and Willett, 1991). While the authors propose several methods to compare such matrices, this approach is not convenient because it does not lead to a fixed size representation of the molecules. Standard vectorial representations can be derived by considering pairs of atoms of the molecule. Topological autocorrelation vectors can for instance be extended to *3D autocorrelation vectors*, computing the autocorrelation of atomic properties from pairs of atoms within a specified Euclidean distance range, instead of a given topological distance on the molecular graph (Wagener et al., 1995). Other representations are based on counting the number of times pairs of atoms of par-

ticular types are found within predefined distance ranges in the 3D structure of the molecule (Carhart et al., 1985; Chen et al., 1998; Swamidass et al., 2005). Considering molecular features based on triplets or larger sets of atoms leads to the notion of *pharmacophore*. A pharmacophore is usually defined as a three-dimensional arrangement of atoms - or groups of atoms - responsible for the biological activity of a drug molecule (Güner, 2000). Typical pharmacophoric features of interest are atoms having particular properties (e.g., positive and negative charges or high hydrophobicity), hydrogen donors and acceptors and aromatic rings centroids (Pickett et al., 1996). In this context, *pharmacophore fingerprints* were proposed as the three-dimensional counterpart of molecular fragment fingerprints. Pharmacophore fingerprints represent a molecule by a bitstring encoding its pharmacophoric content, usually defined as the exhaustive list of triplets of pharmacophoric features found within a set of predefined distances ranges in its 3D structure (McGregor and Muskal, 1999; Brown and Martin, 1996; Matter and Pötter, 1999), although extensions to four-point pharmacophores exist (Mason et al., 1999). Strictly speaking, pharmacophore fingerprints therefore encode *putative* pharmacophores of the molecules, and because the number of potential pharmacophores can be very large, they are usually hashed (Brown and Martin, 1996) or compressed (Abrahamian et al., 2003; Saeh et al., 2005).

A vast amount of descriptors has therefore been proposed in the literature. The above presentation if far from being exhaustive, and we refer interested readers to the textbooks Gasteiger and Engel (2003) and Leach and Gillet (2003) for a detailed presentation. Choosing "good" descriptors for the task to be performed remains nevertheless an open question. For instance, even though the molecular mechanisms responsible for the binding of a ligand to a target are known to strongly depend on their 3D complementarity, different studies account for the superiority of 2D fingerprints over pharmacophore fingerprints in this context (Matter and Pötter, 1999; Brown and Martin, 1996, 1997; Xue et al., 2001b). This observation suggests that 2D fingerprints encode to some extent three-dimensional information (Brown and Martin, 1997), and in many cases, they actually constitute the "gold-standard" representation of chemical structures.

### 1.2.2 Assessing molecular similarity

The notion of similarity plays an important role in chemoinformatics because of a central paradigm, known as the *similarity principle*, stating that molecules of similar structure are likely to exhibit similar biological properties (Johnson and Maggiora, 1990). This principle by itself justifies the methods involving clustering, partitioning and ranking algorithms evoked in Section 1.1.3, and for this reason, the problem of assessing the similarity between molecular structures has been drawing considerable attention in chemoinformatics.

A first class of algorithms considers directly the molecular structures. The similar-

ity between a pair of molecular graphs (2D) can for instance be defined from the ratio between their sizes and the size of their maximal common subgraph (MCS)(Bunke and Shearer, 1998). This approach can be extended to the 3D case by considering the largest set of identical atoms with matching interatomic distances (up to a given resolution) (Pepperrell and Willett, 1991). Measures of similarity between 3D structures can be straightforward obtained by their alignment scores (Thimm et al., 2004; Bultinck et al., 2003; Arakawa et al., 2003). Alternatively, the similarity between aligned molecules can be assessed using the CoMFA methodology, from the comparison of molecular fields by the Carbó similarity index (Carbó et al., 1980), defined for the pair of molecules $(A, B)$ as

$$R_{AB} = \frac{\int\int\int P_A(x,y,z)P_B(x,y,z)dxdydz}{(\int\int\int P_A^2(x,y,z)dxdydz)^{1/2}(\int\int\int P_B^2(x,y,z)dxdydz)^{1/2}},$$

where $P_A$ (resp. $P_B$) is the property measured by CoMFA (e.g., electron density, charge) for the molecule A (resp. B), and the summation is over the elements of the grid surrounding the molecules in the 3D space. In a related approach, molecular shapes can be compared according to the following similarity index

$$S_{AB} = \frac{N}{(T_A T_B)^{1/2}},$$

where $T_A$ (resp. $T_B$) is the number of points of the grid falling inside the Van der Waals volume of the molecule A (resp. B), and $N$ is the number of points falling inside the Van der Waals volume of both molecules (Good and Richards, 1993). Note that this index corresponds to the Carbó similarity index for binary properties $P_A$ and $P_B$ being one at a grid point inside the Van der Waals volume of the molecules A and B respectively, and zero otherwise. However, because searching maximal common subgraphs and aligning molecules are computationally intensive processes, this type of approach suffers from its complexity, and the mainstream approach is to define a similarity measure from the comparison of molecular descriptors.

Because of the great diversity of molecular descriptors, numerous methods were proposed to quantify molecular similarity, based for instance on connectivity indices (Basak et al., 1988), pairs or triplets of atoms in the 2D and 3D representations of molecules (Willett, 1998), matrices of interatomic distances and optimal atom mapping (Pepperrell and Willett, 1991), or autocorrelation vectors (Bauknecht et al., 1996). Because they bear little information about the molecular structures, global physicochemical properties are usually of limited help to quantify the structural similarity of molecules. The most widely used approach to quantify molecular similarity consists in counting the number of shared substructural features. Several *association coefficients* have been defined for that purpose. They are basically based on a dot-product between molecular fingerprints, and mainly differ in the way to normalize the fingerprints overlap (see Willett (1998) for a review). Among these coefficients, the *Tanimoto*

coefficient, defined for the pair of fingerprints $(A, B)$ as

$$T_{AB} = \frac{A^\top B}{A^\top A + B^\top B - A^\top B},$$

has emerged as a standard similarity measure (Salim et al., 2003). This coefficient lies in the $(0, 1)$ range and can be seen as the ratio between the intersection of the fingerprints and their union. Nevertheless, several studies have identified several limits of the Tanimoto coefficient. First, it does not seem to be well adapted to assess the similarity between pharmacophore fingerprints, most probably because of the process of binning inter-atomic distances, which can cause close distances to fall in distinct bins (Matter and Pötter, 1999). Moreover, while it is efficient to detect strong similarity ($T_{AB} > 0.8$) and dissimilarity ($T_{AB} < 0.2$), the values of the Tanimoto coefficient must be subject to caution in the intermediate $(0.2, 0.8)$ range (Flower, 1998). While several attempts have been done to address these issues, with for instance the introduction of fuzzy pharmacophore fingerprints (Horvath and Jeandenans, 2003) and fingerprint scaling (Xue et al., 2001a), efficiently assessing molecular similarity is still an open question in chemoinformatics.

### 1.2.3 Modeling Structure-Activity Relationship

In this section, we consider the task of building a model correlating the biological activity of a molecule with its structure from a pool of compounds of known activity. This model can subsequently be used to predict the activity of unseen compounds in virtual screening applications. From now on, we employ the term activity in a broad sense to refer to a particular biological property the molecules exhibit, such as their ability to bind to a particular biological target, their toxicity, or pharmacokinetics properties. Decades of research in the fields of statistics and machine learning have provided a profusion of methods for that purpose. Their detailed presentation is far beyond the scope of this section, and we invite interested readers to refer to the classical textbooks Duda et al. (2001) and Hastie et al. (2001) for a thorough introduction. In this section we just give general methodological and historical considerations about their application in chemoinformatics.

#### Methodological considerations

Models can be grouped into two main categories depending on the nature of the property to be predicted. Models predicting quantitative properties, such as for instance the degree of binding to a target, are known as *regression* models. On the other hand, *classification* models predict qualitative properties. In SAR analysis, most of the properties considered are in essence quantitative, but the prediction problem is often cast into the binary classification framework by the introduction of a threshold above which the molecules are said to be globally active, and under which globally inactive. In the following, the term classification implicitly stands for binary classification.

In order to build the model, the pool of molecules with known activity is usually split into a *training* set and a *test* set. The training set is used to learn the model. The learning problem consists in constructing a model that is able to predict the biological property on the molecules of the training set, without over-learning it. This *overfitting* phenomenon can for instance be controlled using *cross-validation* techniques, that quantify the ability of the model to predict a subset of the training set that was left out during the learning phase. The test set is used to evaluate the *generalization* of the learned model, corresponding to its ability to make correct prediction on a set of unseen molecules. Different criteria can be used for this evaluation. In regression, it is typically quantified by the correlation between the predicted and the true activity values. In the classification framework, a standard criterion is the accuracy of the classifier, expressed as the fraction of correctly classified compounds. However, if one of the two classes is over-represented in the training set, and/or the cost of misclassification are different, it might be safer to consider the true and false positive and negative rates of classification. The true positive (resp. negative) rate account for the fraction of compounds of the positive (resp. negative) class that are correctly predicted, and the false positive (resp. negative) rate accounts for the fraction of compounds of the negative (resp. positive) class that are misclassified. In virtual screening applications for instance, where we typically do not want to misclassify a potentially active compound, models with low false negative rates are favored, even it they come at the expense of an increased false positive rate.

Because they usually require a limited set of uncorrelated variables as input, applying these models to chemoinformatics requires to summarize the information about the molecules into a limited set of *features*, which may not a trivial task due to the vast amount of molecular descriptors. A popular way to address this problem in chemoinformatics is to rely on *principal component analysis (PCA)* or *partial least squares (PLS)*, that define a limited set of uncorrelated variables from linear combinations of the initial pool of features, in a way to account for most of their informative content. The difference between the two methods stems from the fact that PCA is based on the values of the features only, while PLS considers in addition their correlation with the property to be predicted. Alternatively, *feature selection* methods can be used to identify among an initial pool of features a subset of features relevant with the property to be predicted. Because molecular descriptors are sometimes costly to define, a potential advantage of feature selection methods over PCA- and PLS-based approaches based is the fact that they reduce the number of descriptors to be computed for the prediction of new compounds.

Choosing a model among the profusion of existing models is related to the final goal of the study, and while complex models can for instance have a great predictive ability, this often comes in detriment of their interpretability. We now introduce different methods that have been applied to model SAR.

### Trends in modeling SAR

The first SAR model was developed in 1964 by Hansch and coworkers who applied a multiple linear regression (MLR) analysis to correlate the biological activity of a molecule with a pair of descriptors related to its electronic structure and hydrophobicity (Hansch and Fujita, 1964). MLR models are still widely applied to model SAR. PCA or PLS are commonly used as inputs, in the so-called PC-or PLS- regression models (Saxena and Prathipati, 2003). This is in particular the case in the CoMFA methodology, where PLS regression is the standard model to predict the biological activity of a molecule from the vast amount of grid points sampling the molecular fields (Kubinyi, 2003). Moreover, genetic algorithms have been introduced to perform feature selection as an alternative to standard forward selection or backward elimination approaches (Rogers and Hopfinger, 1994). Related linear approaches can be applied to the classification framework with discriminant analysis algorithms (Martin et al., 1974).

However, because this class of model is limited to encode linear relationships, they can be too restrictive to efficiently predict biological properties. While the models can be enriched with the application of nonlinear transformations of the input variables (Hansch et al., 1968), SAR analysis greatly benefited from the development of nonlinear methods, and in particular artificial neural networks (ANN). Early applications of back-propagation ANN accounted for their predictive superiority over standard linear regression techniques (e.g., Aoyama et al., 1990; Andrea and Kalayeh, 1991; Egolf and Jurs, 1993). Many subsequent studies have demonstrated the strength of ANN to predict biological properties, and they are now a standard tool to model SAR (Devillers, 1996; Zupan and Gasteiger, 1999; Schneider and Wrede, 1998).

Despite their predictive efficiency, a major criticism to ANN is their lack of interpretability, which can be of great importance in chemistry in order to understand the biological mechanisms responsible for the activity. An alternative class of models builds a classifier expressed as a set of rules relating the molecular structure and the biological activity. Such models have been derived for instance using decision trees algorithms (A-Razzak and Glen, 1992; Hawkins et al., 1997) or methods from the field of inductive logic programming (King et al., 1992, 1996), where they provided valuable knowledge about structural features responsible for the activity of molecules.

From the practical viewpoint, another criticism that can be made to ANN is the fact that they require some expertise, concerning for instance the choice of an architecture, in order to be knowledgeably deployed. Moreover, they are known to be prone to overfitting and are hard to reproduce, because of their random initialization and possible convergence to local minima (Manallack and Livingstone, 1999; Burbidge et al., 2001). These theoretical issues are to some extent addressed by the support vector machine (SVM) algorithm, known in particular to avoid the problem of local minima, to prevent overfitting, and to offer a better control of the generalization error (Vapnik, 1998). Moreover, although its good parametrization remains a crucial point, this algorithm requires a less amount of expertise to be deployed. The introduction

of SVM in SAR analysis was pioneered by Burbidge and co-workers (Burbidge et al., 2001). In this study, the SVM algorithm outperforms several ANN architectures for a particular classification task, related to the ability of molecules to inhibit a biological target. Over the last few years, SVM was shown to be a powerful tool for SAR analysis, often outperforming ANN in classification and regression frameworks. We review several applications of SVM to SAR analysis in Section 1.4, but before that, we give in the next section an introduction to the SVM algorithm.

## 1.3 Support Vector Machines and kernel functions

The support vector machine algorithm was initially introduced in 1992 by Vapnik and co-workers in the binary classification framework (Boser et al., 1992; Vapnik, 1998). Over the last decade this method has been gaining considerable attention in the machine learning community, which led to the emergence of a whole family of statistical learning algorithm called kernel methods (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002). SVM has been successfully in many real world applications, including, for instance, optical character recognition (Schölkopf et al., 1995), text-mining (Joachims, 2002) and bioinformatics (Schölkopf et al., 2004), often outperforming state-of-the-art approaches. We start this section by a brief introduction to SVM in the binary classification framework, and, in a second step, we highlight the particular role played by kernel functions in the learning algorithm.

### 1.3.1 SVM for binary classification

In its simplest form SVM is an algorithm to learn a binary classification rule from a set of labeled examples. More formally, suppose one is given a set of examples with a binary label attached to each example, that is, a set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ where $(x_i, y_i) \in \mathcal{X} \times \{-1, +1\}$ for $i = 1, \ldots, \ell$. Here $\mathcal{X}$ is an dot product space (e.g., $\mathbb{R}^d$), equipped with dot product $\langle \cdot, \cdot \rangle$, that represents the space of data to be analyzed, typically molecules represented by $d$-dimensional fingerprints, and the labels $+1$ and $-1$ are meant to represent two classes of objects, such as inhibitors or non-inhibitors of a target of interest. The purpose of SVM is to learn from $\mathcal{S}$ a classification function $f : \mathcal{X} \to \{-1, +1\}$ that can be used to predict the class of new unlabeled examples $x$ as $f(x)$. The classifier output by SVM is based on the sign of a linear function : $f(x) = \text{sign}(\langle w, x \rangle + b)$, for some $(w, b) \in \mathcal{X} \times \mathbb{R}$ defined below. Geometrically, a couple $(w, b) \in \mathcal{X} \times \mathbb{R}$ defines a hyperplane $H_{w,b} = \{x \in \mathcal{X} : \langle w, x \rangle + b = 0\}$ that separates the input space $\mathcal{X}$ into two half-spaces, and the prediction of the class of a new point depends on the position of the point on the one or on the other side of the hyperplane.

#### SVM in the separable case

When the training set is *linearly separable*, i.e., when a hyperplane exists such that the positive and negative examples lie on distinct sides of the hyperplane, the simplest

flavor of SVM, called *hard-margin* SVM, chooses among the infinity of separating hyperplanes the one with the largest distance to the closest data point from $\mathcal{S}$. This distance is known as the *margin* of the hyperplane, usually noted $\gamma$, and this particular hyperplane defines the *maximum margin classifier*, as illustrated in Figure 1.5. The choice of this particular hyperplane intuitively makes sense, but is also well motivated theoretically. SVM learns a classification function that minimizes the empirical risk, while being as regular as possible, and for this reason, it offers an optimal control of the generalization error (Vapnik, 1998).

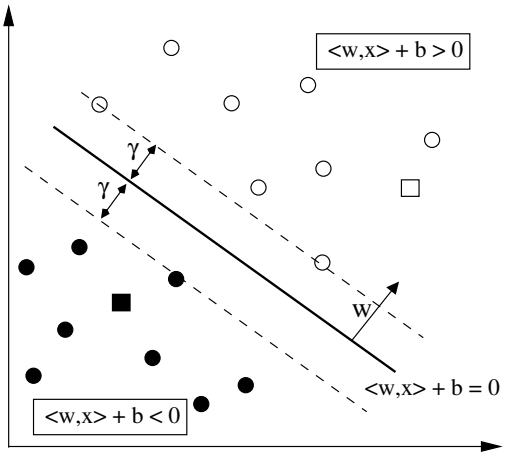


Figure 1.5: SVM finds the hyperplane $\langle w, x\rangle + b = 0$ that separates positive (white circles) and negative (black circles) examples with a maximum margin $\gamma$. Here, a new example represented by the white (resp. black) square is predicted as positive (resp. negative).

The distance from the point $(x_i, y_i) \in \mathcal{S}$ to the hyperplane $H_{w,b} = \{x \in \mathcal{X} : \langle w, x\rangle + b = 0\}$ is given by $|(\langle w, x_i\rangle + b)|/||w||$. If $H_{w,b}$ is a separating hyperplane, this distance is equal to $y_i(\langle w, x_i\rangle + b)/||w||$ and the corresponding margin is given by

$$\gamma = \min_{i=1,\ldots,l} y_i(\langle w, x_i\rangle + b)/||w||,\ (x_i, y_i) \in \mathcal{S}. \tag{1.1}$$

Because hyperplanes are defined up to a scaling constant[4], we can without loss of generality add the following constraint on the definition of $H_{w,b}$:

$$\min_{i=1,\ldots,l} y_i(\langle w, x_i\rangle + b) = 1,\ (x_i, y_i) \in \mathcal{S}. \tag{1.2}$$

This gives a *canonical definition* to hyperplanes with respect to $\mathcal{S}$, and it follows from (1.1) that the margin of a separating hyperplane $H_{w,b}$ is given by $\frac{1}{||w||}$.

[4]Indeed, because $\langle w, x\rangle + b = 0 \Leftrightarrow \langle \alpha w, x\rangle + \alpha b = 0$ for $\alpha \in \mathbb{R}$, $H_{w,b}$ and $H_{\alpha w,\alpha b}$ correspond to the same hyperplane.

The SVM algorithm looks for the canonical separating hyperplane $H_{w^*,b^*}$ having the maximum margin, which can be formulated as the following *optimization problem*:

$$(w^*, b^*) = \underset{\substack{w\in\mathcal{X},\\ b\in\mathbb{R}}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2 \tag{1.3}$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1,\ i = 1, \ldots, l. \tag{1.4}$$

Because the objective function (1.3) and the inequality constraints (1.4) are convex, this problem belongs to the class of *convex problems*. Moreover, because the objective function is strictly convex, this problem admits a unique solution (Boyd and Vandenberghe, 2004). Optimization problems can be solved in practice within the framework of Lagrangian duality. The *Lagrangian* associated to the optimization problem defined by (1.3) and (1.4), denoted in this context as the *primal problem*, is given by

$$L(w, b, \boldsymbol{\alpha}) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i \big(y_i(\langle w, x_i\rangle + b) - 1\big), \tag{1.5}$$

where the coefficients $\alpha_i \geq 0$ are known as the *Lagrange multipliers* associated to the constraints $y_i(\langle w, x_i\rangle + b) \geq 1$, and we let $\boldsymbol{\alpha} \in \mathbb{R}^l$ be the vector of Lagrange multipliers. The unique solution of this optimization problem is given at the *saddle point* of the Lagrangian, corresponding to a minimum of $L(w, b, \boldsymbol{\alpha})$ with respect to $(w, b)$, and a maximum with respect to $\boldsymbol{\alpha}$. We first minimize the Lagrangian with respect to $w$ and $b$ for a fixed $\boldsymbol{\alpha}$. This is done by setting the partial derivatives $\frac{\partial}{\partial w}L(w, b, \boldsymbol{\alpha})$ and $\frac{\partial}{\partial b}L(w, b, \boldsymbol{\alpha})$ to zero, leading respectively to

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{1.6}$$

and

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{1.7}$$

Substituting (1.6) and (1.7) into the Lagrangian (1.5), the original problem therefore becomes equivalent to the following *dual problem*

$$\boldsymbol{\alpha}^* = \ \operatorname{argmax}_{\boldsymbol{\alpha}\in\mathbb{R}^l} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \tag{1.8}$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, l \tag{1.9}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{1.10}$$

This problem is a *quadratic program* that can be solved efficiently in practice using a dedicated algorithm, known as *Sequential Minimal Optimization (SMO)* (Platt, 1999),

taking advantage of the constraints (1.9) and (1.10).

When the optimum $\boldsymbol{\alpha}^*$ is met, the decision function $f(x) = \text{sign}(\langle w^*, x\rangle + b^*)$ writes as

$$f(x) = \text{sign}(\sum_{i=1}^{l} \alpha_i^* y_i \langle x, x_i\rangle + b^*). \tag{1.11}$$

The offset $b^*$ can be obtained by the fact that at the optimum, the points verify the following condition, known as the *Karush-Kuhn-Tucker condition*:

$$\alpha_i^* \big(y_i(\langle w^*, x_i\rangle + b^*) - 1\big) = 0, \quad i = 1, \ldots, l.$$

A striking consequence of this condition is that, from (1.4), non-zero Lagrange multipliers $\alpha_i^*$ can only be associated to points $(x_i, y_i)$ for which $y_i(\langle w^*, x_i\rangle + b^*) = 1$. These points lie on the margin of the hyperplane and, importantly, they are the only ones to enter its definition. For this reason, they are called the *Support Vectors* of the classifier. The offset $b^*$ can therefore be obtained from a support vector $(x_i, y_i)$ as

$$b^* = \frac{y_i}{\langle w^*, x_i\rangle} = \frac{y_i}{\sum_{j=1}^{l} \alpha_j^* \langle x_j, x_i\rangle},$$

but, in practice, it is better to average this quantity over the whole set of support vectors (Burges, 1998).

We note finally that by varying the value of the offset, we can tune the false positive and negative rates of the SVM classifier. For instance, considering an offset $b < b^*$ will have the effect of translating the decision function towards the negative side of the hyperplane, thereby favoring the classification of test data as positive in the test phase. While this would have the effect of increasing the false positive rate, that is, the classification as positive of points belonging to the negative class, it is also likely to reduce the false negative rate, which can be important in applications such as virtual screening where we typically do not want to miss a potentially active compound. Actually, gradually increasing the offset from $-\infty$ to $+\infty$ defines a generic SVM classifier based on the vector $w^*$, from which we can draw the evolution of the true positive versus the false negative rates in a curve known as the *Receiver Operating Characteristic (ROC)* curve. Not only can the ROC curve help choosing the value of the offset achieving the optimal trade-off between the true positive and the false positive rates, but the area under this curve defines an indicator of the performance of this generic SVM classifier, known as the AUC, that is considered to be more general than the accuracy of the single SVM classifier defined by $b^*$, and is gaining an increasing popularity in the machine learning community (Fawcett, 2003). The AUC of an ideal classifier would be 1, meaning that the positive data would be the first to be recognized as positive according to their scores, while for a random classifier it would be 0.5.

**SVM in the non separable case**

The above algorithm applies to separable data, but in the general case, a linear hyperplane separating the data may not exist. Non separable datasets can be considered in practice by the introduction of *slack-variables* $\xi_i \geq 0$ for each point $(x_i, y_i) \in \mathcal{S}$ in order to relax the separability constraints (1.4) of the primal problem by

$$y_i(\langle w, x_i \rangle) + b \geq 1 - \xi_i.$$

The term $\xi_i$ accounts for the error made in the classification of the point $(x_i, y_i)$: if $\xi_i = 0$, the point $(x_i, y_i)$ is correctly classified and lies outside the margin, if $0 < \xi_i \leq 1$ it is still correctly classified but lies within the margin, and if $\xi_i > 1$, the point is misclassified. This is illustrated in Figure 1.6. Note that the margin of the hyperplane is defined from the points $(x_i, y_i) \in \mathcal{S}$ for which $\xi_i = 0$.

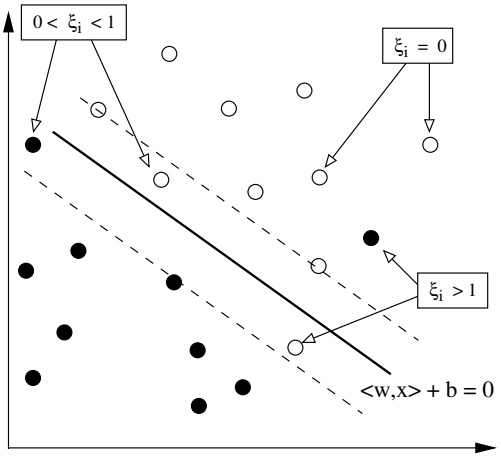


Figure 1.6: Illustration of the introduction of slack-variables. $\xi_i > 1$ corresponds to misclassified points, $0 < \xi_i < 1$ corresponds to points correctly classified within the margin, and $\xi_i = 0$ corresponds to points correctly classified outside the margin.

The sum of the slack variable therefore accounts for the amount of separability constraints violation on the training set and must therefore be controlled during the learning process. This is done in practice by adding this quantity to the objective function of the SVM optimization problem, which becomes

$$(w^*, b^*, \boldsymbol{\xi}^*) = \operatorname{argmin}_{\substack{w \in \mathcal{X}, \\ b \in \mathbb{R}, \\ \boldsymbol{\xi} \in \mathbb{R}^l}} \quad \frac{1}{2}||w||^2 + C \sum_{i=1}^{l} \xi_i \tag{1.12}$$

$$\text{s.t.} \quad y_i(\langle w, x_i \rangle) + b \geq 1 - \xi_i, \quad i = 1, \ldots, l \tag{1.13}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, l. \tag{1.14}$$

The constant $C$ introduced in the objective function (1.12) is meant to introduce a trade-off between the maximization of the margin, expressed by the term $\frac{1}{2}||w||^2$, and the separation of the training set, expressed by the sum of the slack variables. Decreasing $C$ has the effect of tolerating a larger number of misclassifications on the training set for the profit of an increased margin. This formulation is known as the *soft-margin* SVM, in opposition to the hard-margin formulation that tolerates no misclassifications in a separable case. Note that the hard-margin formulation is retrieved if $C = +\infty$, and even though the training set happens to be linearly separable, this soft-margin formulation prevents overfitting to noise or outliers.

Although we skip details, the soft margin hyperplane defined by (1.12), (1.13) and (1.14) is obtained by a slight modification of the dual problem derived in the separable case, resuming in turning the constraints $\alpha_i \geq 0$ of equation (1.9) into $0 \leq \alpha_i \leq C$, for $i = 1, \ldots, l$. Finally, note that several soft-margin formulations have been proposed depending on the way to penalize the violation of the separability constraints. For instance an alternative formulation considers the squared values of the slack-variables in (1.14) (Shawe-Taylor and Cristianini, 2004).

### Non-linear SVM

Nevertheless, when dealing with nonlinearly separable problems, such as the one depicted on Figure 1.7 (left), the set of linear classifiers may not be rich enough to provide a good classification function, no matter what the values of the parameters $w \in \mathcal{X}$, $b \in \mathbb{R}$, and possibly $\boldsymbol{\xi} \in \mathbb{R}^l$, are. SVMs can be straightforward generalized to the nonlinear case by applying a linear approach to the transformed data $\phi(x_1), \ldots, \phi(x_\ell)$ rather than the original data, where $\phi$ is a nonlinear embedding from the input space $\mathcal{X}$ to a *feature space* $\mathcal{H}$ usually, but not necessarily, of higher dimensionality, equipped with dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Replacing the dot products $\langle x_i, x_j \rangle$ by $\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$ in (1.8) computes a linear hyperplane in the feature space, that, according to (1.11), corresponds to the following nonlinear separating function $f$ in the input space

$$f(x) = \text{sign}\left( \sum_{i=1}^{\ell} \alpha_i \langle \phi(x_i), \phi(x) \rangle_{\mathcal{H}} + b \right) . \tag{1.15}$$

A classical example is given in Figure 1.7, where the natural ellipsoidal separating function in the input space $\mathcal{X} = \mathbb{R}^2$ (left) corresponds to an hyperplane in the feature space $\mathcal{H} = \mathbb{R}^3$ (right) defined by the mapping

$$\begin{aligned} \phi : \quad \mathbb{R}^2 &\to \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \end{aligned} \tag{1.16}$$

This is due to the fact that ellipses can be written as linear equations of $z_1$ and $z_2$, which also explains that the corresponding hyperplane is parallel to $z_3$.

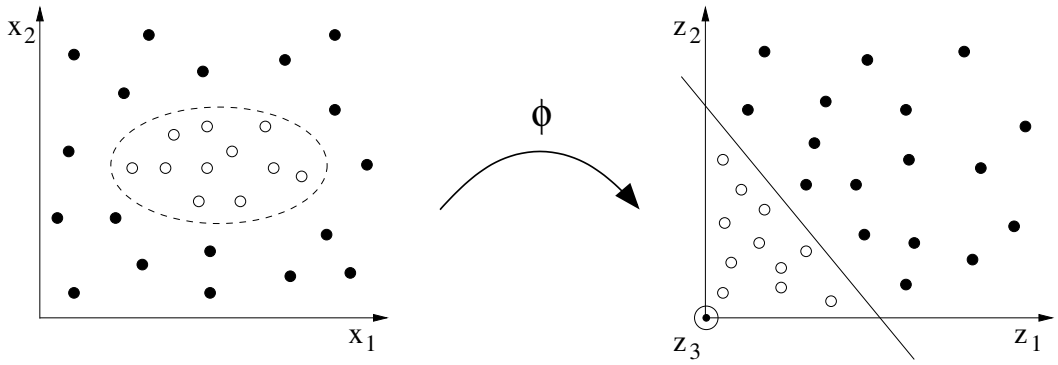


Figure 1.7: Illustration of non-linear SVMs, where the mapping $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is defined by $\phi([x_1, x_2]) = [z_1, z_2, z_3]$, where $z_1 = x_1^2$, $z_2 = x_2^2$ and $z_3 = \sqrt{2}x_1x_2$.

### 1.3.2 Kernel functions

An important remark is that in the dual formulation of SVM, the data are only present through dot products: pairwise dot products between the training points during the learning phase (1.8), and dot products between a new data and the training points during the test phase (1.11). This means that instead of explicitly knowing $\phi(x)$ for any $x \in \mathcal{X}$, it suffices to be able to compute dot products $\langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ for any $x, x' \in \mathcal{X}$. A striking point is that in some cases, it is more efficient to compute directly the dot product without explicitly computing the mapping $\phi$. If we consider for instance the mapping $\phi$ given in equation (1.16), it is easy to see that, for $(x, y) \in \mathcal{X}$, the dot product $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ is given by $\langle x, y \rangle^2$, which almost reduces the cost of evaluating the dot product in $\mathcal{H}$ by a factor three[5].

A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ computing directly the dot product in the feature space is known as a *kernel function*. Standard kernel functions for vectorial input spaces $\mathcal{X} = \mathbb{R}^n$ include:

- the *polynomial* kernel $K(x, y) = (\langle x, y \rangle + R)^d$. This kernel implicitly maps a point $x = [x_1, \ldots, x_n]$ to a feature space indexed by all the products of monomials $x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$ such that $\sum_{j=1}^n i_j \leq d$, and computes a dot product in this space (Shawe-Taylor and Cristianini, 2004). Note that the polynomial kernel generalizes the mapping of equation (1.16) associated to the kernel $K(x, y) = \langle x, y \rangle^2$.

- the *radial basis function (RBF)* kernel $K(x, y) = \exp(-\frac{||x-y||^2}{2\sigma^2})$. This kernel computes a dot product in a feature space of infinite dimensions. The resulting decision function writes as a sum of Gaussian functions centered on the support vectors. Note that the smaller $\sigma$, the more peaked the Gaussian functions, and,

[5]Indeed, computing explicitly $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ requires 11 sum and product operations, while only 4 are needed to compute $\langle x, y \rangle^2$.

for a sufficiently small value of $\sigma$, this kernel can always perfectly separate the training set (Shawe-Taylor and Cristianini, 2004).

These two kernels map nonlinearly the input data into high- (even infinite-) dimensional feature spaces at almost no cost, and are known to be safe default choices for SVMs. Note that the dot product in the input space $\mathcal{X}$ is known as the *linear* kernel.

SVM offer the possibility to consider such high-dimensional feature space for two reasons. The first theoretical reason stems from the fact that the learning algorithm is able to deal with high dimensionality by a heavy use of regularization (Vapnik, 1998). The second practical reason is due to the fact that complex feature spaces can be considered implicitly by the use of kernel functions. This latter mathematical trick, which is common to the whole family of kernel methods (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002), is known as the *kernel trick* and is considerably enriched by the following definition.

**Definition 1 (Positive definite kernel).** *Let $\mathcal{X}$ be a nonempty space. Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function. $K$ is said to be a positive definite kernel if and only if, for all positive integer $\ell$, for all $x_1, \ldots, x_\ell \in \mathcal{X}$, the square $\ell \times \ell$ matrix $\mathbf{K} = (K(x_i, x_j))_{1 \leq i,j \leq \ell}$ is positive semi-definite, that is, all its eigenvalues are nonnegative.*

For a given set $\mathcal{S}_x = \{x_1, \ldots, x_\ell\}$, $\mathbf{K}$ is the *Gram matrix* of $K$ with respect to $\mathcal{S}_x$. A fundamental property of positive definite kernels is the fact that each such kernel can be represented as a dot product in some space. More precisely, it can be shown (Aronszajn, 1950) that for any positive definite kernel function $K$, there exists a space $\mathcal{H}$, equipped with the dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that:

$$\forall u, v \in \mathcal{X}, \quad K(u, v) = \langle \phi(u), \phi(v) \rangle_{\mathcal{H}} \,. \tag{1.17}$$

The kernel trick consists in replacing all occurrences of $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ by a positive definite kernel $K$, which turns out to be equivalent to transforming the input patterns $x_1, \ldots, x_\ell$ into the corresponding vectors $\phi(x_1), \ldots, \phi(x_\ell) \in \mathcal{H}$ and to look for hyperplanes in $\mathcal{H}$, the striking point being that we do not even need to know explicitly the nature of the mapping $\phi$. Note from equation (1.8) that the knowledge of the Gram matrix suffices to obtain the coefficients $\alpha_i$, and the decision function $f$ of equation (1.15) is now given for an input pattern $x$ by:

$$f(x) = \operatorname{sign} \left( \sum_{i=1}^{\ell} \alpha_i K(x_i, x) + b \right) \,. \tag{1.18}$$

Positive definite kernels are therefore the key ingredient of SVM and, more generally, of kernel methods. Because they correspond to a dot product in a particular feature space, they can intuitively be seen as measures of similarity between the input data. Actually, in some cases, it may be easier to directly compare objects than

choosing an explicit representation, and as a consequence of the kernel trick, any prior notion of similarity on the input space can be used as input to SVM provided it verifies the positive definiteness condition. This allows in particular to generalize SVM to non-vectorial spaces $\mathcal{X}$ provided a proper kernel function exists to assess the similarity between patterns of the input space. For instance, SVMs have recently been applied in the fields of bioinformatics, natural language processing and chemoinformatics, with the introduction of kernels for DNA sequences and proteins primary structures (Schölkopf et al., 2004), parse trees (Collins and Duffy, 2001) and 2D and 3D structures of molecules (Kashima et al., 2004; Swamidass et al., 2005; Mahé et al., 2006).

On the practical side, a noteworthy feature of support vector machines, and more generally of kernel methods, is that, since ready-to-use libraries to derive separating hyperplanes are available [6], the only requirement for them to be applied to a specific problem is to derive a proper kernel.

## 1.4 Support Vector Machines for virtual screening

In this section, we present several applications of SVM to structure-activity relationship and virtual screening. In a first step, we consider the mainstream vector-based approach where the molecules are represented by a set of molecular descriptors. In a second step, we present the alternative approach that consists in defining a kernel function comparing directly the molecular structures, without the need of explicitly extracting molecular descriptors.

### 1.4.1 SVM and molecular descriptors

Following the pioneer introduction of SVM for SAR by Burbidge et al. (2001), SVMs have been increasingly used in SAR and virtual screening based on a great variety of molecular descriptors, in order to solve various tasks that we now review[7].

#### Prediction of inhibition activity

In their pioneer study, Burbidge et al. (2001) consider the task of predicting the ability of pyrimidines molecules to inhibit the dihydrofolate reductase enzyme. The original regression problem, involving the prediction of a quantitative measure of inhibition, is cast to a binary classification problem, which consists in predicting if the molecule A has a higher activity that the molecule B, for each pair (A,B) of molecules of the data set. SVM is shown to outperform decision trees, three artificial neural networks

[6] The website `http://www.kernel-machines.org/` gathers many resources related to SVM and kernel methods for instance.

[7] Note however that this review is by no means exhaustive.

and a radial basis function network for this purpose. Saeh et al. (2005) and Yao et al. (2004) use SVM for the prediction of GPCR and COX-2 inhibitors, using respectively pharmacophore fingeprints and a combination of constitutional, topological and electrostatic descriptors. The reported models are accurate, outperforming multiple linear regression and radial basis function networks in the latter case. Byvatov and Schneider (2004) and Liu (2004) consider the problem of feature selection in several tasks involving the detection of kinase, thrombin and factor Xa inhibitors. The main conclusion is that SVM performs better when all features are used, but in Byvatov and Schneider (2004) the number of features could be reduced by a factor 10 with only a slight decrease in accuracy of the model, which is beneficial in terms of computation times in the testing phase. Wilton et al. (2003) and Jorissen and Gilson (2005) evaluate SVM for ranking molecules according to their biological activity. In both studies, molecules are simply ranked according to their classification score. While Wilton et al. (2003) suggest that SVM is less efficient than simple methods based on molecular fingerprints and the Tanimoto coefficient, Jorissen and Gilson (2005) show that SVM can compare favorably to these approaches if the parameters of the model are chosen to optimize a novel ranking-based criterion. Finally, SVM have been used in particular machine learning frameworks. Weston et al. (2003) consider transductive SVM associated to an heuristic feature selection scheme to address a classification problem strongly unbalanced, and in high-dimensions, consisting in the prediction of thrombin inhibitors. Using the same dataset, Warmuth et al. (2003) use SVM in an active learning framework, with a view to reproducing a real case application where the molecules to be screened have to be selected iteratively.

### Drug/non-drug classification

In a problem consisting in the discrimination of known drugs and random molecules, SVM is shown to be at least comparable, and often better, than models based on ANN, linear discriminants, bagged decision trees and bagged $k$-nearest neighbors (Byvatov et al., 2003; Müller et al., 2005). In Takaoka et al. (2003), SVM and artificial neural networks show comparable results for the identification of drug-like compounds, according to a particular criterion defined by chemists'intuition and ease of synthesis,. Finally, Zernov et al. (2003) use SVM to predict drug-likeliness and agrochemical-likeliness for large sets of molecules (30 000 and 11 000), and report better results than artificial neural networks.

### Prediction of pharmacokinetics and toxicity properties

In Sorich et al. (2003), SVM is used in a classification framework to discriminate between substrates or non-substrates of a family of enzymes involved in the metabolism of drugs, and is shown to outperform, in general, partial least squares discriminant analysis and Bayesian regularized artificial neural networks. Together with a recursive feature elimination scheme (Guyon et al., 2002), SVM is applied to a similar prob-

lem involving the identification of substrates of a family of proteins able to transport molecules across the plasma membrane, which constitutes for instance a major cause of failures of cancer chemotherapy (Xue et al., 2004e), and to predict pharmacokinetic and toxicological properties of molecules (Xue et al., 2004d). Feature selection reduces the number of features used to represent the molecules, which can be helpful to identify features responsible for toxicity and poor pharmacokinetics properties (Xue et al., 2004d), and the results obtained by SVM are slightly better than those obtained with $k$-nearest neighbors, decision trees and probabilistic neural networks (Xue et al., 2004e). In Yap and Chen (2005), SVM are involved in the identification of inhibitors and substrates of three enzymes known to have an important effect on drug metabolism. In this study, molecules are classified according to consensus classification schemes, based on several SVMs trained on different sets of descriptors selected by a genetic algorithm. This consensus approach leads to results comparable with those of earlier studies, and is shown to be better than a single SVM classification system. Alternatively, SVM regression has been used to predict the toxicity of phenol molecules (Yao et al., 2004), the binding affinity to human serum albumin, a protein known to have an important role in the distribution of drugs in the organism (Xue et al., 2004c) and the aqueous solubility of molecules, which is a limiting factor to the absorption of drugs (Lind and Maltseva, 2003). In these three studies, SVM is systematically shown to outperform models based on multiple linear regression and/or artificial neural networks.

**Prediction of general molecular properties**

Finally, SVM regression has been used to predict general molecular properties such as, for instance, the capacity factor of peptides (Liu et al., 2004), the O-H bond dissociation energy of phenols (Xue et al., 2004b) and the heat capacity of molecules (Xue et al., 2004a). In these studies, SVM is often shown to outperform multiple linear regression models and several artificial neural networks.

In addition to these approaches based on standard molecular descriptors, SVM has been used based on vectorial representations of molecules derived from *graph mining* algorithms. Graph mining consists in the identification of interesting structural patterns within a set of graphs. A molecule can then be represented by a vector indexed by these identified patterns. This approach is therefore related to the structural keys characterization of molecules introduced in Section 1.2.1, with the key feature that the patterns relevant with the task to be performed are identified automatically. For instance, in a binary classification framework, Kramer et al. (2002) and Helma et al. (2004) automatically extract from molecular graphs a set of *discriminative* linear fragments: linear molecular fragments appearing frequently in one class, but only rarely in the other. Alternatively, Deshpande and Karypis (2002) propose to characterize molecular graphs by general subgraphs occurring frequently in the data set. An important difference with the above approaches lies in the fact that, here, the frequency

criterion is global, and is not balanced between high frequency in one class and low frequency in the other. Because this approach is obviously likely to detect insignificant subgraphs, in particular if the set of inactive compounds is larger than the set of active compounds, the authors suggest to rely on subgraphs frequently occurring in the class of active compounds only. This approach was recently extended to the 3D case in Deshpande et al. (2005). In this study, 3D structures of molecules are characterized by frequent *geometrical* subgraphs, defined as topological (2D) subgraphs mapped in the 3D space by the introduction of atomic coordinates. Discovering frequent geometrical subgraphs consists in the identification of frequent topological subgraphs, that have a similar 3D structure. In practice, the tridimensional information is dealt by the introduction of a global shape criterion characterizing geometrical subgraphs. Frequent geometrical subgraphs then define as frequent topological subgraphs that have the same shape criterion value, up to a given resolution. The conjunction of graph mining approach with SVMs led to promising results. Indeed, the resulting models are in general able to efficiently predict biological properties, and at the same time, using a linear kernel, this formulation offers the possibility to identify structural features responsible for activity (Helma et al., 2004; Deshpande et al., 2005).

### 1.4.2 SVM and kernel functions for molecules

Developing kernels for molecules is part of the larger thematic of kernel design for *structured data* that do not have a natural vectorial representation, such as strings, trees or graphs. This topic has been drawing considerable interest over the recent years in the machine learning community, motivated by many real world applications in several fields such as bioinformatics, natural language processing, or chemical informatics among others. The mainstream approach to tackle this problem is a divide-to-conquer approach consisting in deriving a kernel for structured objects from sub-kernels comparing their parts. A general framework was early proposed for this purpose with the introduction of *convolution kernels* (Haussler, 1999). Convolution kernels rely on the idea that a structured object $x \in \mathcal{X}$ can be decomposed into a $d$-tuple of substructures $(x_1, \ldots, x_d) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. A string $s$ can for instance be decomposed into a tuple of $d$ substrings by $s = s_1 \circ s_2 \circ \cdots \circ s_d$, where $\circ$ denotes the string concatenation operator. This notion of decomposition can be defined formally by a binary relation $\mathcal{R}$ on $\mathcal{X} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$, $\mathcal{R}(x, \vec{x})$ being one if $x$ can be decomposed into $\vec{x} = (x_1, \ldots, x_d)$, and zero otherwise. The set of possible decompositions of an object $x$ is then defined by $\mathcal{R}^{-1}(x) = \{\vec{x} : \mathcal{R}(x, \vec{x}) = 1\}$. The idea of the convolution kernel is to sum, over the allowed decompositions of the objects to be compared, the product of sub-kernels comparing pairs of elements of their decomposition. This can be interpreted as a convolution operation, and the kernel introduced in Haussler (1999) writes as

$$K(x, y) = \sum_{\vec{x} \in \mathcal{R}^{-1}(x)} \sum_{\vec{y} \in \mathcal{R}^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d) \tag{1.19}$$

In practice, this general formulation may require considerable work to make the convolution kernel applicable to a particular problem. Several issues related in particular to the number of allowed decompositions, the ease of detecting the substructures considered and the computational cost of the sub-kernels, are critical for the computability of the kernel. For this reason, most kernels for structured data correspond to particular convolution kernels, where the generality of the formulation was simplified for the sake of computability.

An important example is that of *spectral kernels*. Spectral kernels consider the frequency of occurrence of a predefined set of substructures $\mathcal{S}$ to characterize a structured object. In their simplest form, they write as standard dot products $K(x, y) = \langle \phi(x), \phi(y) \rangle$ where $\phi(x)$ is the vector counting the substructures in $x$, defined as $\phi(x) = \big(\phi_s(x)\big)_{s \in \mathcal{S}}$, where $\phi_s(x)$ is the number of times the substructure $s$ appears in $x$. If we let $p(x)$ denote the set of parts of the object $x$, spectral kernels correspond in the context of convolution kernels to a decomposition relation $\mathcal{R}(x, p(x))$ of order $d = 1$, being one if there exists a structure $s \in \mathcal{S}$ such that $p(x) = s$, and a binary kernel between substructures $K_S(u, v) = \mathbf{1}(u = v)$, being one if the elements $u$ and $v$ are identical. Spectral kernels can be written as a special case of convolution kernel according to the following equation

$$K(x, y) = \sum_{u \in \mathcal{R}^{-1}(x)} \sum_{v \in \mathcal{R}^{-1}(y)} \mathbf{1}(u = v)$$

Spectral kernels have been developed for instance:

- in the context of strings, where the $k$-spectrum kernel is based on the count of common substrings of length $k$ (Leslie et al., 2002). Computing the kernel therefore requires the extraction and matching of all contiguous substrings of length $k$ from the strings to be compared. Several refinements have been subsequently introduced in order to handle more flexible string characterizations, such as tolerance to gaps in the detection of the substrings, and mismatches in their comparison (Leslie et al., 2004).

- in the context of labeled trees (Collins and Duffy, 2001) based on the count of common subtrees. This kernel implicitly maps trees to a feature space indexed by the whole set of labeled trees defined by the tree labeling alphabet, and computes a dot product in this space.

Despite the fact that these kernels map the objects to feature spaces of high dimensionality, they can be computed efficiently, using suffix tree algorithms in the former case and dynamic programming in the latter case. The striking point is that even though the feature space can be very large, the number of substructures to be be detected depends on the size of the objects to be compared.

In the context of convolution and spectral kernels, several kernels have been proposed more recently to compare *labeled graphs* A labeled graph $G$ is defined by a set of vertices $\mathcal{V}_G$, a set of edges $\mathcal{E}_G \subset \mathcal{V}_G \times \mathcal{V}_G$ connecting pairs of vertices, and a labeling function $l$, that assigns a label taken from an alphabet $\mathcal{A}$ to vertices and edges. Graph kernels find natural applications in chemoinformatics where molecular graphs representing covalent bonds between atoms are naturally represented as labeled graphs. Following the spectral approach, one would therefore like to represent graphs by their exhaustive list of subgraphs, and define a graph kernel from the count of common subgraphs. However, because the number of subgraphs increases exponentially with the size of the graphs, and their comparison involves to detect graph isomorphism, this kernel cannot be computed in polynomial time (Gärtner et al., 2003). As a result, pioneer graph kernels involve a restricted set of subgraphs and characterize graphs simple linear subgraphs (Kashima et al., 2003; Gärtner, 2002). Put in the context of convolution kernels, the corresponding decomposition relation writes as

$$\mathcal{R}\big(G, (v_0, \ldots, v_n)\big) = 1 \text{ if } \begin{cases} v_i \in \mathcal{V}_G, \ i = 0, \ldots, n \\ (v_i, v_{i+1}) \in \mathcal{E}_G, \ i = 0, \ldots, n-1 \end{cases}, \ \forall n \in \mathbb{N}$$

and this class of graph kernels can be written as

$$K(G, G') = \sum_{u \in \mathcal{R}^{-1}(G)} \sum_{u' \in \mathcal{R}^{-1}(G')} K_S(u, u'), \tag{1.20}$$

where $K_S$ is a kernel between subgraphs. This class of kernels characterizes a graph by an infinite number of substructures, and in practice, the subgraph kernels must be down-weighted to ensure that their summation converges.

In the graph theory terminology, the set of allowed decompositions of the graph $G$, $\mathcal{R}^{-1}(G)$, corresponds to its set of *walks*. From now on, we adopt this terminology, and we refer to the class of graph kernels defined by equation (1.20) as *walk-based* graph kernels. We let $\mathcal{W}(G)$ be the set of walks of the graph $G$, defined formally as $\mathcal{W}(G) = \cup_{n=0}^{\infty} \mathcal{W}_n(G)$, where $\mathcal{W}_n(G) = \{(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1} : (v_i, v_{i+1}) \in \mathcal{E}_G, i = 0, \ldots, n-1\}$ is the set of walks of length $n$[8]. Moreover, we extend the graph labeling function to label walks, a walk label being naturally defined as the concatenation of the labels of the vertices and edges it is made of. Consequently, we rewrite equation (1.20) as

$$K(G, G') = \sum_{w \in \mathcal{W}(G)} \sum_{w' \in \mathcal{W}(G')} K_{\mathcal{W}}(w, w'), \tag{1.21}$$

where $K_{\mathcal{W}}$ is a *walk kernel*, typically a string kernel assessing the similarity between pairs of walks from their labels. We now review the different walk-based kernels that have been introduced.

[8]Note that the length of a walk is defined as the number of edges it is made of.

**Kernels based on common walks**

A first class of walk-based graph kernels involves the count of common walks and is therefore related to the spectral formulation. Gärtner (2002) first introduced a *label pairs* graph kernel based on the count of walks having the same starting and ending label. The corresponding walk kernel is formally defined as

$$K_{\mathcal{W}}\big((v_0,\dots,v_n),(v'_0,\dots,v'_m)\big) = \lambda_m\lambda_n \mathbf{1}\big(l(v_0) = l(v'_0) \wedge (l(v_n) = l(v'_m)\big),$$

where the parameters $\lambda_i$ give weights to the walks depending on their length. The associated feature space is indexed by pairs of vertex labels, and if we let $\mathcal{A}_v = \{l_1,\dots,l_{|\mathcal{A}|}\}$ be the set of vertex labels, the label pairs kernel can be written as a standard dot product:

$$K(G,G') = \sum_{i=1}^{|\mathcal{A}_v|}\sum_{j=1}^{|\mathcal{A}_v|} \phi_{l_i,l_j}(G)\phi_{l_i,l_j}(G'),$$

where

$$\phi_{l_i,l_j}(G) = \sum_{n=0}^{+\infty} \lambda_n |\{(v_0,\dots,v_n) \in \mathcal{W}_n(G) : l(v_0) = l_i \wedge l(v_n) = l_j\}|.$$

Although we skip details, this kernel can be computed in polynomial times for well chosen values of the parameters $\lambda_i$. In particular, choosing $\lambda_i = \gamma^i$ for a sufficiently small value of $\gamma$, or $\lambda_i = \beta^i/i!$, leads respectively to the *geometric* and *exponential* graph kernel. However, despite the fact that an infinite number of walks is considered to represent the graphs, the dimensionality of the feature space associated to the kernel is equal to $|\mathcal{A}_v|^2$, which is likely to be too small to efficiently assess precise graph similarity, in particular when the set of vertex labels is small. Moreover, while it might be relevant to allow gaps in the comparison of walks, the fact that the kernel does not take into account the information along the walks can be questioned.

In order to enrich the expressiveness of the feature space, Gärtner et al. (2003) introduced a *contiguous label sequences* graph kernel. This kernel is based on the count of globally identical walks, and the corresponding walk kernels writes as

$$K_{\mathcal{W}}(w,w') = \lambda_{|w|}\mathbf{1}(l(w) = l(w')),$$

where $\mathbf{1}(l(w) = l(w'))$ is one if the walks $w$ and $w'$ are identically labeled, meaning that they have the same length and identical sequences of vertex and edge labels, and zero otherwise. The corresponding feature space has an infinite number of dimensions, indexed by the set $\mathcal{S}$ of possible walk labels that can be obtained from the alphabet $\mathcal{A}$, and the kernel writes as a standard dot product:

$$K(G,G') = \langle \phi(G), \phi(G') \rangle,$$

where

$$\phi(G) = \big(\phi_s(G)\big)_{s\in\mathcal{S}} \text{ and } \phi_s(G) = \sqrt{\lambda_{|s|}}|\{w \in \mathcal{W}_{|s|}(G) : l(w) = s\}|.$$

This kernel can also be computed in polynomial times for well chosen values of the parameters $\lambda_i$. The algorithm consists in merging the graphs $G$ and $G'$ into their *product graph* $G \times G'$ , the striking point being that a bijection can be found between $\mathcal{W}(G\times G')$ and the pairs of walks identically labeled taken from $\mathcal{W}(G)$ and $\mathcal{W}(G')$. We postpone the details of the computation to Chapter 2. Finally, note that this kernel was further refined in order to handle mismatches in the comparison of walk labels (Gärtner et al., 2003).

Because they are based on binary walk kernels, this first class of walk-based graph kernels can be viewed as special cases of spectral kernels. However, because the number of walks found in a graph can be infinite, in the case of a graph with cycles for instance, the walk counts must be down-weighted by parameters $\lambda_i$, that typically decrease with the length of the walks. It is worth noting that even though an infinity of features is taken into account to represent the graphs, well chosen values of these parameters lead to closed-form computations of the kernels.

### Marginalized kernels

A related approach was proposed in Kashima et al. (2003) in the context of marginalized kernels. This kernel is also based on the comparison of walks, but it differs from the above walk-counts formulation in two aspects. First, walks are considered as random walks and are associated to a probability of occurring in the graphs. Second, the formulation adopted is more general in the sense that it allows to match differently labeled walks. More precisely, following the general definition of equation (1.21), the walk kernel $K_{\mathcal{W}}$ writes as

$$K_{\mathcal{W}}(w, w') = p_G(w)p_{G'}(w')K_{\mathcal{L}}(w, w'), \tag{1.22}$$

where $p_G$ is a probability distribution on $\mathcal{W}(G)$, and $K_{\mathcal{L}}$ is a kernel between walk labels. In Kashima et al. (2004), the kernel $K_{\mathcal{L}}$ is defined as a product of kernels $K_{\mathcal{L}_\mathcal{V}}$ and $K_{\mathcal{L}_\mathcal{E}}$, namely,

$$\begin{aligned} K_{\mathcal{L}}\big((v_0,\ldots,v_n),(v'_0,\ldots,v'_m)\big) = \mathbf{1}(n=m) & \\ \times \prod_{i=0}^{n} K_{\mathcal{L}_\mathcal{V}}\big(l(v_i), l(v'_i)\big) & \\ \times \prod_{i=0}^{n-1} K_{\mathcal{L}_\mathcal{E}}\big(l((v_i,v_{i+1})), l(v'_i,v'_{i+1})\big), & \end{aligned}$$

where the kernels $K_{\mathcal{L}_\mathcal{V}}$ and $K_{\mathcal{L}_\mathcal{E}}$ respectively compare pairs of vertex and edge labels. Moreover, the probability distribution $p_G$ is defined as a first order Markov model, that is,

$$p_G((v_o, \ldots, v_n)) = p_s^{(G)}(v_0) \prod_{i=0}^{n-1} p_t^{(G)}(v_{i+1}|v_i),$$

where $p_s^{(G)}$ and $p_t^{(G)}$ are respectively initial and transition probabilities chosen such that $\sum_{w \in \mathcal{W}(G)} p_G(w) = 1$. Note that because the kernel $K_\mathcal{L}$ between walk labels allows the matching of differently labeled walks, the marginalized kernel does not have an explicit interpretation as a feature vector, in opposition to the above walk-counts graph kernels. Nevertheless, because of the closure properties of the class of kernel functions, the above construction is known to be valid as long as the basis kernels $K_{\mathcal{L}_\mathcal{V}}$ and $K_{\mathcal{L}_\mathcal{E}}$ are proper kernels.

However, in the absence of prior information on the similarity of vertex and edge labels, a natural choice for the kernels $K_{\mathcal{L}_\mathcal{V}}$ and $K_{\mathcal{L}_\mathcal{E}}$ is a binary kernel, checking whether the labels are identical or not. In this case the walk kernel writes as

$$K_\mathcal{W}(w, w') = p_G(w) p_{G'}(w') \mathbf{1}(l(w) = l(w')),$$

where $\mathbf{1}(l(w) = l(w'))$ is one if the walks have the same label, meaning that they have the same length and identical sequences of vertex and edge labels, and zero otherwise. Under this parametrization, the marginalized graph kernel can be written as a standard dot product $K(G, G') = \langle \phi(G), \phi(G') \rangle$ in an infinite dimensional feature space indexed by the set $\mathcal{S}$ of possible walk labels, defined by $\phi(G) = \big(\phi_s(G)\big)_{s \in \mathcal{S}}$, and $\phi_s(G) = \sum_{w \in \mathcal{W}(G)} p_G(w) \mathbf{1}(l(w) = s)$. This formulation therefore bears strong similarities with the contiguous label sequences graph kernel (Gärtner et al., 2003), at the difference that walks are here weighted by their probability of occurring in the graph, instead of a factor decreasing with their length. From this consideration, it was shown that the product graph formalism, initially introduced to count identically labeled walks, can be extended to compute the marginalized graph kernel in its general formulation (Kashima et al., 2004). We will come back to this in Chapter 2.

In the context of molecular graphs, because labeled walks correspond to the definition of molecular fragments, we can note a strong similarity between the class of walk-based graph kernels and the hashed fingerprint characterization of molecules introduced in Section 1.2.1. There are however important differences in the sense that fragments of infinite length are considered in these kernels, no hashing process is required, and fragments are weighted by a continuous parameter encoding its frequency or probability of occurrence in the molecule, whereas fingerprints simply encodes their presence or absence.

### More kernels

Note that according to the definition of walks, vertices are allowed to appear several times in a walk. While this representation might look restrictive, walk-based graph kernels gave promising results in chemoinformatics (Kashima et al., 2004; Mahé et al., 2005) and bioinformatics (Borgwardt et al., 2005; Karklin et al., 2005) applications, often comparing to state-of-the-art approaches. However, Ramon and Gärtner (2003) highlight the limited expressiveness of walk-based graph kernels, showing in particular that different graphs can be mapped to the same point in the corresponding feature space. With the view to increase their expressivity, one would therefore rather consider *paths*, that is, walks made of distinct vertices, in the above kernel constructions. Unfortunately, such kernels were shown to be not computable in polynomial times (Gärtner et al., 2003). Ramon and Gärtner (2003) therefore suggest that the expressivity of graph kernels must be traded for their computability. As a first step toward the refinement of the feature space corresponding to walk-based graph kernels, a kernel based on the count of common subtrees is introduced. Its implementation is based on a dynamic programming algorithm that recursively extends tuples of vertices neighbors. While the complexity over the walk-based kernels is increased, the authors suggest it might be affordable if the connectivity of the graphs is limited and the there is a sufficient diversity in the labels of the vertices. This is likely to be the case of molecular graphs, but the proposed kernel was not validated experimentally, at least to our knowledge. Following these considerations, several graph kernels have been proposed that overcome the limitations of the walk-based characterization of graphs.

Horváth et al. (2004) introduce a graph kernel based on the detection of *cyclic* and *tree patterns*. These patterns are defined as canonical representations of cycles and trees, being in particular unique up to isomorphism. To detect such patterns, a graph is first split into a set of cycles[9], and a set of trees resulting from the removal of the edges belonging to the extracted cycles. The kernel is then defined as the sum of the common cyclic and tree patterns that can be found in decomposed graphs. Note importantly that the kernel is defined as the number of different *types* of patterns that can be found in the graphs, which means that the frequency of the cycles or subtrees corresponding to particular patterns is not taken into account in the kernel. In the general case, this kernel cannot be computed in polynomial times, but it can be obtained efficiently by the introduction of an upper bond on the admissible number of cycles in the graphs. This boils down to assuming a certain kind of "well-behavedness" of the data, which seems to be verified in practice in the case of molecular graphs. Indeed, 99.76% of the 42689 molecules of the NCI dataset, on which was validated the kernel, have less than 100 cycles. This kernel is shown experimentally to significantly outperform approaches based on the discovery of frequent subgraphs.

Borgwardt and Kriegel (2005) propose a kernel based on the count of common paths. However, because it is not possible to consider exhaustive sets of paths, the kernel construction is restricted to the sets of shortest paths between pairs of vertices.

[9] A cycle of the graph $G$ is a path $v_0, \ldots, v_n$, that is, $v_i \neq v_j, 0 \leq i, j \leq n$, where $v_0 = v_n$.

Importantly, an edge-based characterization of paths is adopted, where a path is defined as a succession of distinct edges. As a consequence, vertices can appear several times in such paths, which contrasts with the classical definition of paths as walks made of distinct vertices. The kernel definition follows that of convolution kernels, and is expressed as the sum of a kernel between paths, over all pairs of shortest paths found in the graphs. For a given pair of paths, the path kernel writes as the product of a kernel between their starting and ending vertices, and a kernel comparing their length. This kernel can be computed more efficiently than the kernels based on exhaustive sets of walks, and on a task involving classification of proteins, they led to significantly better results.

Motivated by chemical applications, other graph kernels more specific to molecular graphs have been introduced. In Ralaivola et al. (2005) a family of graph kernels based on *generalized fingerprints* is introduced. A molecular graph $G$ is represented by a vector $\phi(G)$ indexed by a set $\mathcal{S}$ of molecular fragments, that is, sequences of atom types and covalent bond types up to a given length, typically set to 8 or 10. In the standard fingerprint characterization of molecules, $\phi(G)$ corresponds to a hashed bitstring which size ranges from 512 to 2048. This hashed representation is compared to a classical vectorial representation defined as $\phi(G) = (\phi_s(G))_{s\in\mathcal{S}}$, where $\phi_s(G)$ can either be one if the fragment $s$ occurs in the molecule, or count the number of times it appears. The generalization of fingerprints is therefore two fold in the sense that first, the vector $\phi(G)$ is not necessarily hashed, and second, it is not necessarily a bitstring. Note that in practice the generalized fingerprints need not be stored explicitly. By analogy with the Tanimoto coefficient introduced in Section 1.2.2, the proposed kernels formulate as variations of the *Tanimoto kernel* defined as

$$K(G, G') = \frac{\langle \phi(G), \phi(G') \rangle}{\langle \phi(G), \phi(G) \rangle + \langle \phi(G'), \phi(G') \rangle - \langle \phi(G), \phi(G') \rangle}.$$

The proposed kernels often reach state of the art results on tasks involving the classification of molecules, and experiments suggest that generalized fingerprints are more efficient to characterize molecular structures than usual hashed fingerprints[10]. Finally, it is worth noting that while this family of kernels bears a strong similarity with walk-based graph kernels, their implementation follows a completely different approach. Indeed, because the length of the fragments considered is bonded, they can explicitly be detected by depth first search algorithms. Although we skip details, this has important consequences in practice because first, the proposed kernels can be computed efficiently using an algorithm derived from that of spectrum string kernels (Leslie et al., 2002), and second, it allows to consider paths or cycles instead of general walks.

In Fröhlich et al. (2005) an *optimal assignment kernel* is introduced, based on the idea of optimally assigning the atoms from one molecule to those of another. The kernel formulates as a sum of kernels between pairs of atoms, which has to be maximized over

[10]Note that the classical *Daylight* fingerprints were used as a baseline in the study.

all possible assignment of the set of atoms of the smaller molecule to the set of atoms of the bigger one. The kernel comparing pairs of atoms follows a similar construction: it is defined as a sum of kernels between atoms' neighbors, which has to be maximized over the possible assignments of the set of neighbors of one atom to the set of neighbors of the other. This construction can be further extended to take into account the similarity between remote neighbors (i.e., by considering neighbors of neighbors, and so on) into the kernel between atoms. Because the degree of graph vertices is bonded by four in molecular graphs, the global kernel computation can be carried out by a greedy procedure, in a polynomial time comparable to that of the kernels based on exhaustive sets of walks. Experiments show that this kernel sometimes led to better results than the marginalized kernel Kashima et al. (2004), but unfortunately, it is not positive definite.

Finally, although not directly related to molecular graphs, several kernel functions are proposed in Swamidass et al. (2005) to compare 1D and 3D structures of molecules. The 1D structure is defined as the *SMILES* representation of molecules, a string encoding the atomic constitution and covalent bonds of molecules that is widely used in chemoinformatics. Classical string kernels (Leslie et al., 2002, 2004) are then applied to compare these SMILES representations. The 3D structure is encoded by a set of histograms of inter-atomic distances. One histogram corresponds to a particular pair of atom types (e.g., Carbon/Oxygen), and a fixed-size vectorial representation can be defined from the concatenation of the histograms. Such vectors are then compared by a standard RBF kernel. Despite the simplicity of the representation, 1D kernels provide results that are only slightly below those obtained with generalized fingerprints and Tanimoto kernels [11]. On the other hand, the kernel between 3D structures consistently led to the worst results.

## 1.5 Contribution of the thesis

In this thesis, we consider the problem of defining kernel functions for molecular structures and their applications in virtual screening using SVM. This section gives a quick overview of our contributions toward this goal, and draws the outline of the following chapters.

### 1.5.1 Extensions of marginalized graph kernels

The pioneer graph kernels (Kashima et al., 2004; Gärtner et al., 2003) based on the comparison of common walks in the graphs are subject to several problems in practice. First, their computational complexity is cubic with respect to the product of the sizes of the two graphs to be compared, which results in slow implementation for real-world

[11] Note, these kernels were initially introduced in Ralaivola et al. (2005), but are put in a broader context in this paper.

problems, and limits in particular their applicability to virtual screening where the databases to be screened typically involve thousands of molecules. Moreover, it might not be optimal to characterize a graph by its exhaustive set of walks for at least two reasons: on the one hand, some walks may contain relatively little information (e.g., a sequence $C - C - C$), while on the other hand, many walks are irrelevant because they represent "tottering walks" on the graph, that is, walks which return to a visited vertex immediately after leaving it.

In the first part of the thesis, we introduce two extensions of the original formulation of the marginalized kernel between labeled graph (Kashima et al., 2004), 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 fragments of such labels contain information about the environment of each atom, and computation time, because the number of identical fragments between two molecules significantly decreases. Second, we show how to modify the random walk model proposed in Kashima et al. (2004) in order to remove totters, without increasing the complexity of the implementation.

### 1.5.2 Tree-pattern graph kernels for 2D structures

While they led to promising results in the fields of chemoinformatics (Kashima et al., 2004; Mahé et al., 2005) and bioinformatics (Borgwardt et al., 2005; Karklin et al., 2005) for instance, Ramon and Gärtner (2003) highlighted the limited expressiveness of walk-based graph kernels, showing in particular that many different graphs can be mapped to the same point in the corresponding feature space. As a first step toward a refinement of the feature space representation of graphs, Ramon and Gärtner (2003) introduced a kernel function comparing graphs on the basis of their common subtrees. While this representation looks promising, the proposed kernel was not deeply analyzed nor evaluated experimentally on real world applications, at least to our knowledge.

We propose in the second part of the thesis to validate the relevance of such tree features to represent molecules with graph kernels. In a first step we revisit the formulation introduced by Ramon and Gärtner (2003), from which we derive two kernels with explicit feature spaces and inner products. One parameter entering their definition allows to gradually increase the complexity of the features considered to represent the graphs. Decreasing the parameter results in a classical walk-based kernel, and by moving this parameter, we can observe in detail the effect of increasing the number and the complexity of the tree features representing the graphs. When the size of allowed subtrees is increased however, we show that the practical use of this kernel is limited by the explosion in the number of subtrees occurring in the graphs. In a second step, we therefore introduce two extensions to the initial formulation of the kernels that allow, on the one hand, to extend and generalize their associated feature space, and on the other hand, to prevent a set of noisy patterns to be detected in the graphs.

### 1.5.3 Pharmacophore kernels for 3D structures

It is widely accepted that several drug-like properties can be efficiently deduced from the 2D structure of the molecule, the Lipinski's "rule of five" remaining a widely used standard for the prediction of intestinal absorption (Lipinski et al., 2001), and the prediction of mutagenicity from 2D molecular fragments being an accurate state-of-the-art approach (King et al., 1996). In the case of target binding prediction, however, 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.

Motivated by this consideration, we introduce in the last part of the thesis a family of positive definite kernels specifically optimized for the manipulation of 3D structures of molecules. The kernels are based on the comparison of the three-points pharmacophores present in the 3D structures of molecules, a set of molecular features known to be particularly relevant for virtual screening applications. We present a computationally demanding exact implementation of these kernels, as well as fast approximations related to the classical fingerprint-based approaches.

### 1.5.4 ChemCpp

Note finally that the different kernels presented in this thesis have been implemented within the open-source *ChemCpp* toolbox, a C++ toolbox dedicated to the computation of kernel functions for molecular structures. Its development was initiated by Jean-Luc Perret while he was a postdoctoral fellow in the Bioinformatics Center of Kyoto University (Japan), and is available at `http://chemcpp.sourceforge.net`.

Chapter 2

# Extensions of marginalized graph kernels

This work appeared in a slightly different form in the *Journal of Chemical Information and Modeling*, co-authored with Nobuhisa Ueda, Jean-Luc Perret, Tatsuya Akutsu and Jean-Philippe Vert (Mahé et al., 2005).

## Introduction

Kashima et al. (2004); Gärtner et al. (2003) recently introduced positive definite kernels between labeled graphs, based on the detection of their common walks. 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. Together with the support vector machine algorithm, they offer the possibility to model structure activity relationships (SAR) of molecules from their 2D structure, without the need for explicit molecular descriptors computation. Despite their promising results in virtual screening applications (Kashima et al., 2004), they are subject to several problems in practice. First, their computational complexity is cubic with respect to the product of the sizes of the two graphs to be compared, which results in slow implementations for real-world problems, and limits in particular their applicability to virtual screening where the databases to be screened typically involve thousands of molecules. Moreover, it might not be optimal to characterize a graph by its exhaustive set of walks for at least two reasons: on the one hand, some walks may contain relatively little information (e.g. a sequence $C - C - C$), while on the other hand, many walks are irrelevant because they represent "tottering walks" on the graph, that is, walks which return to a visited vertex immediately after leaving it.

In this chapter, we introduce two extensions of the original formulation of the marginalized kernel between labeled graph (Kashima et al., 2004), 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 fragments of such labels contain information

about the environment of each atom, and computation time, because the number of identical fragments between two molecules significantly decreases. Second, we show how to modify the random walk model proposed in Kashima et al. (2004) in order to remove totters, without increasing the complexity of the implementation. Experiments on two mutagenicity datasets validate the proposed extensions, making this approach a possible complementary alternative to other SAR modeling strategies. This chapter is organized as follows. We introduce notations and review the marginalized kernel between labeled graphs in Section 2.1. The pair of proposed extensions are presented in Sections 2.2 and 2.3, and are validated experimentally in Section 2.4. finally, Section 2.5 gives concluding remarks.

## 2.1 Marginalized graph kernels

In this section we define the basic notations and briefly review the graph kernels introduced in Kashima et al. (2004) and Gärtner et al. (2003), upon which are based the extensions that will be presented in Sections 2.2 and 2.3.

### 2.1.1 Labeled directed graphs

A *labeled graph* $G = (\mathcal{V}_G, \mathcal{E}_G)$ is defined by a finite set of *vertices* $\mathcal{V}_G$, a set of *edges* $\mathcal{E}_G \subset \mathcal{V}_G \times \mathcal{V}_G$, and a labeling function $l : \mathcal{V}_G \cup \mathcal{E}_G \to \mathcal{A}$ which assigns a *label* $l(x)$ taken from an alphabet $\mathcal{A}$ to any vertex or edge $x$. We let $|\mathcal{V}_G|$ be the number of vertices of $G$, $|\mathcal{E}_G|$ be its number of edges. In *directed* graphs, edges are oriented and to each vertex $u \in \mathcal{V}_G$ corresponds a set of *incoming neighbors* $\delta^-(u) = \{v \in \mathcal{V}_G : (v, u) \in \mathcal{E}_G\}$ and *outgoing neighbors* $\delta^+(u) = \{v \in \mathcal{V}_G : (u, v) \in \mathcal{E}_G\}$. We let $d^-(v) = |\delta^-(u)|$ be the *in-degree* of the vertex $u$, and $d^+(v) = |\delta^+(u)|$ be its *out-degree*.

We assume below that a set of labels $\mathcal{A}$ has been fixed, and we represent a molecular graph by a labeled directed graph $G = (\mathcal{V}_G, \mathcal{E}_G)$. To do so, we let the set of vertices $\mathcal{V}_G$ correspond to the set of atoms of the molecule, the set of edges $\mathcal{E}_G$ to its covalent bonds, and label these graph elements according to an alphabet $\mathcal{A}$ consisting of the different types of atoms and bonds. Note that since graphs are directed, a pair of edges of opposite direction is introduced for each covalent bond of the molecule. Figure 2.1 shows a chemical compound seen as a labeled directed graph.

For a given graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, we let $\mathcal{V}_G^* = \cup_{n=1}^{\infty} \mathcal{V}_G^n$ be the set of finite-length sequences of vertices. A *walk* $h$ of length $n$ in the graph $G$ is a sequence of $n+1$ vertices $(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1}$ with the property that $(v_i, v_{i+1}) \in \mathcal{E}_G$ for $0 \leq i \leq n-1$. We note $|w|$ the length of the walk $w$, and we let $\mathcal{W}(G) \subset \mathcal{V}_G^*$ be the set of walks of the graph $G$, defined formally as $\mathcal{W}(G) = \cup_{n=0}^{\infty} \mathcal{W}_n(G)$, where $\mathcal{W}_n(G) = \{(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1} : (v_i, v_{i+1}) \in \mathcal{E}_G, i = 0, \ldots, n-1\}$ is the set of walks of length $n$. The labeling function $l : \mathcal{V}_G \cup \mathcal{E}_G \to \mathcal{A}$ can be extended as a function $l : \mathcal{W}(G) \to \mathcal{A}^*$, where $\mathcal{A}^*$ is the set of finite-length sequences of labels taken from $\mathcal{A}$, the label $l(w)$ of a walk $w = (v_0, \ldots, v_n) \in \mathcal{W}(G)$ being naturally defined as the succession of the labels of the

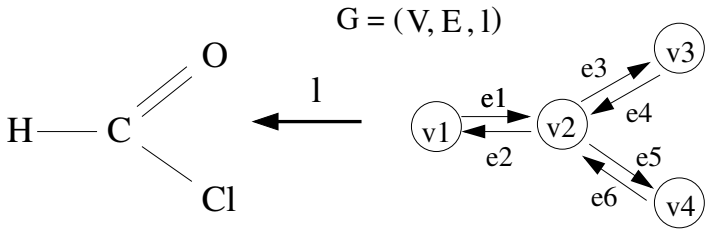


Figure 2.1: A chemical compound seen as a labeled graph

vertices and edges it is made of: $l(w) = (l(v_0), l(v_0, v_1), l(v_1), \ldots, l(v_{n-1}, v_n), l(v_n)) \in \mathcal{A}^{2n+1}$.

### 2.1.2 Marginalized graph kernels

The kernel we consider in this work was introduced in Kashima et al. (2004), based on the general *marginalized kernels* formulation (Tsuda et al., 2002). Marginalized kernels define a global similarity measure by means of a simpler one, expressed on a set of latent variables associated to the input data. In our case, these latent variables consist of the set of walks of the graphs, which are easier to handle than general subgraphs (Gärtner et al., 2003). Walks are assumed to be generated by a random walk process on the graphs, and the kernel between two graphs is then defined as the expectation of the pairwise walks similarity, according to their probability distributions. In labeled graphs, sequences of vertex and edge labels are associated to the walks of the graph. Their similarity can therefore assessed by a string kernel, and the kernel introduced in Kashima et al. (2004) boils down to the following formula :

$$K(G_1, G_2) = \sum_{\substack{(w_1, w_2) \\ \in \mathcal{V}_{G_1}^* \times \mathcal{V}_{G_2}^*}} p_{G_1}(w_1) p_{G_2}(w_2) K_{\mathcal{L}}\left(l(w_1), l(w_2)\right), \tag{2.1}$$

where $p_{G_1}$ and $p_{G_2}$ are probability distributions on $\mathcal{V}_{G_1}^*$ and $\mathcal{V}_{G_2}^*$, and the function $K_{\mathcal{L}} : \mathcal{A}^* \times \mathcal{A}^* \mapsto \mathbb{R}$ is a string kernel between label sequences.

Kashima et al. (2004) focus on the particular case where the kernel $K_{\mathcal{L}}$ in (2.1) is the Dirac function $\delta$ :

$$K_{\mathcal{L}}(l_1, l_2) = \delta(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

thus accounting for a perfect similarity between walks if they share the same label and null otherwise[1]; and where, for a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, the probability $p_G$ on $\mathcal{V}_G^*$

[1]Note that if two label sequences are identical they implicitly have the same length.

factorizes as a first order random walk model :

$$p_G\left((v_0,\ldots,v_n)\right) = p_s(v_0)\prod_{i=1}^{n} p_t(v_i|v_{i-1}). \tag{2.3}$$

In order to ensure that (2.3) defines a probability distribution on $\mathcal{V}_G^*$ (i.e., $\sum_{w\in\mathcal{V}_G^*} p_G(w) = 1$), we must impose constraints on the emission and transition probabilities $p_s$ and $p_t$. This can be done, for example, by choosing parameters $0 < p_q(v) < 1$ for $v \in \mathcal{V}_G$, an initial probability distribution $p_0$ on $\mathcal{V}_G$ ($\sum_{v\in\mathcal{V}_G} p_0(v) = 1$), a transition matrix $p_a$ on $\mathcal{V}_G \times \mathcal{V}_G$ ($\sum_{v\in\mathcal{V}_G} p_a(v|u) = 1$ for $u \in \mathcal{V}_G$) positive only along edges ($p_a(v|u) > 0 \Rightarrow (u,v) \in \mathcal{E}_G$), and by setting, for any $u, v \in \mathcal{V}_G^2$,

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

Under these conditions it can easily be checked that (2.3) is a probability distribution on $\mathcal{V}_G^*$ 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 walks have positive probabilities under $p$: $p_G(w) > 0 \Rightarrow w \in \mathcal{W}(G)$. Figure 2.2 shows an example of this particular probabilistic model.

Following the definition introduced in Helma et al. (2004), we let $\mathcal{S}(\mathcal{A})$ be the set of *linear molecular fragments* taken from $\mathcal{A}$, that is, the set of sequences of bonds-connected atoms based on the labels of $\mathcal{A}$. For a given fragment $s \in \mathcal{S}(\mathcal{A})$ we introduce a mapping $\phi_s : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ denotes the space of labeled directed graphs, defined for a given graph $G$ as $\phi_s(G) = \sum_{w\in\mathcal{W}(G)} p_G(w)\delta(l(w), s)$. If we let $K_\mathcal{L}$ be the Dirac kernel, the kernel (2.1) can be written as a standard dot-product based on the molecular fragments $\mathcal{S}(\mathcal{A})$:

$$\begin{aligned} K(G_1, G_2) &= \langle \Phi(G_1), \Phi(G_2) \rangle \\ &= \sum_{s\in\mathcal{S}(A)} \phi_s(G_1)\phi_s(G_2) \end{aligned}$$

Under this parametrization, the kernel (2.1) therefore maps the graphs into an infinite-dimensional space where each dimension corresponds to a particular linear fragment. This shows a strong analogy with the widely used hashed fingerprints characterization of molecules introduced in Section 1.2.1. There are however several important differences with the approach illustrated above. First, while marginalized kernels take into account every single molecular fragment to compare the molecules, which is equivalent to dealing with an infinite-dimensional feature vector, fingerprints only involve a smaller number of features[2]. Moreover, because they only require their dot product,

[2]Typically the set of molecular fragments up to length 7 or 8.

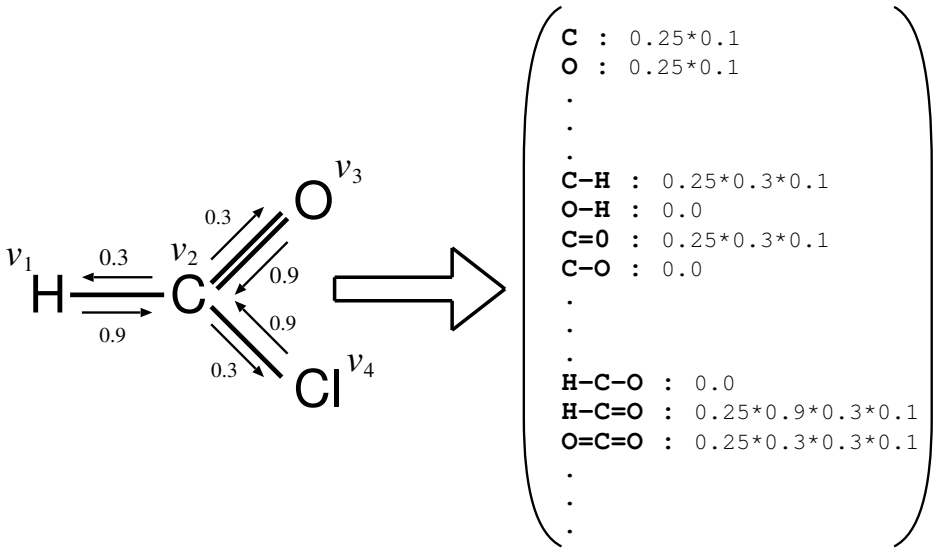


Figure 2.2: A molecular graph $G$ (left) and its feature-space representation $\phi(G)$ (right). Here, $\forall\ v_i \in \mathcal{V}_G$, $p_q(v_i) = 0.1$, $p_a(v_j|v_i) = 1/d^+(v_i)$ iff $(v_i, v_j) \in \mathcal{E}_G$, and $p_0$ can be chosen to be the uniform distribution, i.e. $p_0(v_i) = 1/|\mathcal{V}_G| = 0.25$. The values $(1 - p_q(v_i))p_a(v_j|v_i)$ are shown along each edge $(v_i, v_j)$ of the graph. (note that $\sum_{v_j \in \mathcal{V}_G} p_a(v_j|v_i) = 1,\ \forall\ v_i \in \mathcal{V}_G$). For the walk $w = (v_1, v_2, v_3)$, we therefore have $l(w) = (H, -, C, =, O)$ and $p(w) = 0.25 * 0.9 * 0.3 * 0.1$. The right hand side of the picture shows such examples of walks possibly occurring in the graph, together with their associated probabilities.

marginalized kernels don't need to explicitly store the high dimensional feature vectors $\phi(G_1)$ and $\phi(G_2)$, thereby bypassing the clashing phenomenon occurring with hashed fingerprints. Another important difference lies in the way of dealing with the substructures. Instead of just checking the presence of molecular fragments, marginalized kernels can quantify their occurrence in the graph according to probability distributions. This approach offers a more flexible way to evaluate the influence of the substructures in the graph similarity. Note that the related method presented in Gärtner et al. (2003) is equivalent to defining an infinite-dimensional fingerprint counting the frequency of appearance of the molecular fragments in the graphs.

### 2.1.3 Kernels computation

While the kernel definition (2.1) involves a summation over an infinite number of walks, it can be computed efficiently using product graphs and matrix inversions introduced in Gärtner et al. (2003), and briefly recalled below.

Given two labeled graphs $G_1 = (\mathcal{V}_{G_1}, \mathcal{E}_{G_1})$ and $G_2 = (\mathcal{V}_{G_2}, \mathcal{E}_{G_2})$, their *product graph*

is defined as the labeled graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ whose vertices $\mathcal{V}_\mathcal{G} \subset \mathcal{V}_{G_1} \times \mathcal{V}_{G_2}$ are pairs of vertices with identical labels ($(v_1, v_2) \in \mathcal{V}_\mathcal{G}$ iff $l(v_1) = l(v_2)$), and an edge connects the vertices $(u_1, u_2)$ and $(v_1, v_2)$ iff $(u_i, v_i) \in \mathcal{E}_{G_i}$, for $i = 1, 2$, and $l(u_1, v_1) = l(u_2, v_2)$.

Let us now define a functional $\pi$ on the set of walks $\mathcal{W}(\mathcal{G})$ by

$$\pi\left((u_0, v_0), (u_1, v_1), \ldots, (u_n, v_n)\right) = \pi_s\left(u_0, v_0\right) \prod_{i=1}^{n} \pi_t\left((u_i, v_i) | (u_{i-1}, v_{i-1})\right),$$

with

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

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 walks in (2.3) on the graph $G_1$ (resp. $G_2$).

If the label kernel $K_\mathcal{L}$ is chosen to be the Dirac kernel (2.2), then the kernel (2.1) only involves walks that can be found concurrently in the two graphs. By construction of the product graph, there is a bijection between this set of common walks and the set of walks $\mathcal{W}(\mathcal{G})$ of the product graph. Using the definition of the functional $\pi$, it can then be shown that:

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

Define now the $|\mathcal{V}_\mathcal{G}| \times |\mathcal{V}_\mathcal{G}|$ transition matrix $\Pi_t = \left(\pi_t(v|u)\right)_{(u,v) \in \mathcal{V}_\mathcal{G}^2}$. Walks in the product graph can be generated by raising this matrix to a particular power. If one now defines the $|\mathcal{V}_\mathcal{G}|$-dimensional vector $\pi_s = \left(\pi_s(v)\right)_{v \in \mathcal{V}_\mathcal{G}}$, it can be checked that:

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

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

$$\begin{aligned} K(G_1, G_2) &= \sum_{n=0}^{\infty} \left( \sum_{\substack{w \in \mathcal{W}(\mathcal{G}), \\ |w| = n}} \pi(w) \right) \\ &= \pi_s^\top \left(I - \Pi_t\right)^{-1} \mathbf{1}. \end{aligned}$$

The direct computation of a matrix inversion has a complexity cubic in the size of the matrix. In the case of the above product graph, the size of the matrix $\Pi_t$ is at worst $|\mathcal{V}_{G_1}| \times |\mathcal{V}_{G_2}|$, and this approach can be time consuming. However, this matrix is typically sparse, and savings can be achieved using an approximation of the matrix inverse based on the first terms of its power series expansion : $(I - \Pi_t)^{-1} \approx \sum_{i=0}^{N} \Pi_t^i$. Generally speaking, if we note $|M|$ the number of non-zero elements of a matrix $M$,

and $d(M)$ its maximum number of non-zero elements per line, computing the product of two $(n \times n)$ sparse matrices $A$ and $B$ has a complexity of $O(|A|d(B))$. Moreover, if we note $d = d(A)$, we have $d(A^k) \leq \min(d^k, n)$. From these two observations, it follows that computing the sum $\sum_{i=0}^{N} A^i$ has a complexity of $O\left(|A| \sum_{i=1}^{N-1} \min(d^i, n)\right)$ (Smola and Kondor, 2003) . Note that if no hypothesis is made about the value of $d$, this complexity reduces to $O(|A|nN)$. By construction of the product graph $\mathcal{G} = G_1 \times G_2$, we have $|\mathcal{V}_\mathcal{G}| \leq |\mathcal{V}_{G_1}| \times |\mathcal{V}_{G_2}|$ and $|\mathcal{E}_\mathcal{G}| \leq |\mathcal{E}_{G_1}| \times |\mathcal{E}_{G_2}|$. Moreover, if $d_1^+$ and $d_2^+$ are the maximum out-degrees of the nodes of $G_1$ and $G_2$, it follows that the maximum out-degree of the nodes of the graph $\mathcal{G}$ is less or equal than $d_1^+ d_2^+$. This means that the size of the matrix $\Pi_t$ is bounded by $|\mathcal{V}_{G_1}| \times |\mathcal{V}_{G_2}|$, its maximum number of non-zero elements by $|\mathcal{E}_{G_1}| \times |\mathcal{E}_{G_2}|$, and its maximum non-zero elements per line by $d_1^+ d_2^+$. It therefore follows that the approximation of the matrix $(I - \Pi_t)^{-1}$ by the first $N$ terms of its power series expansion has a complexity of $O\left(|\mathcal{E}_{G_1}||\mathcal{E}_{G_2}| \sum_{i=1}^{N-1} \min((d_1^+ d_2^+)^i, |\mathcal{V}_{G_1}||\mathcal{V}_{G_2}|)\right)$.

In the case where many vertices have identical labels, the product graph used to compute the graph kernel has many vertices too, since the number of vertices in the product graph corresponds to the number of pairs of vertices with identical labels. As a result, the computation of the graph kernel can be time consuming, and this method may be difficult to use on large chemical data-banks involving several hundred thousand molecules. As an example, the computation can take several hundred milliseconds on a recent desktop computer to compute the kernel between two chemical compounds with a moderate number of atoms (typically between 10 and 50). Moreover, one might expect the search of common walk labels to be too naive to detect interesting patterns between chemical compounds. These two points constitute important issues to tackle in order to use this type of graph kernels in real-world applications. We now present two modifications of the original kernel with the goals to increase its relevance as a similarity measure between molecular compounds, usually denoted as its *expressive* power, and to reduce its computational complexity.

## 2.2 Label enrichment with the Morgan index

One possibility to address both issues simultaneously is to increase the specificity of labels, for example 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 walks 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 walks are replaced by walks labeled with their environment.

For the kind of applications we focus on in this paper—classification of chemical compounds—it seems natural to consider the chemical environment of atoms. For instance it makes sense to distinguish between atoms with similar labels but that belong

to different functional groups. As a first attempt to define such local environment, we propose to introduce information related to the topological environment of the vertices in the labeling function of the graphs. To do so, we compute for each vertex of the graph an index called the *Morgan index* (Morgan, 1965) , that is defined by a simple iterative procedure. Initially, the Morgan indices are equal to 1 for every vertex. Then, at each iteration, the Morgan index of each vertex is defined as the sum of the Morgan indices of its adjacent vertices. Mathematically, if we let $M_i$ be the vector of the Morgan indices computed at the $i$th iteration, this reads $M_0 = \mathbf{1}$ and $M_{n+1} = A_{dj}M_n$, where $A_{dj}$ is the graph adjacency matrix and $\mathbf{1}$ the unity vector. This process is illustrated in Figure 2.3.

The Morgan index was initially developed to determine canonical representations of molecules, and is considered a good and fast solution to detect graph isomorphism. Note moreover that the Morgan index associated to a particular vertex after $n$ iterations actually counts the number of walks of length $n$ that start in that vertex and end somewhere in the graph. This vertex descriptor has already been studied in chemical graph theory, and is known as the *atomic length-n walk-count* descriptor in the literature(Rücker and Rücker, 1993) .

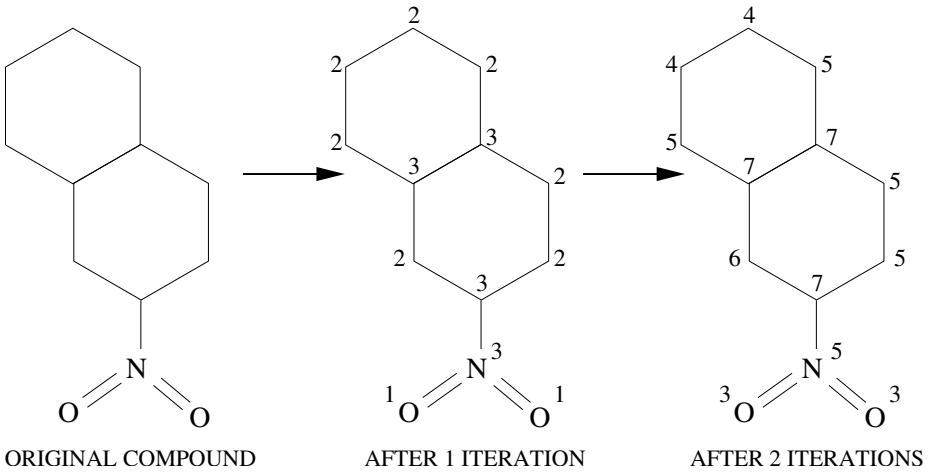


Figure 2.3: Morgan index process

Finally, given the Morgan indices after $n$ iterations, we propose to augment the label of a vertex by its value, before computing the marginalized graph kernel. This results in a family of kernels $(K_n)_{n\geq 0}$, indexed by the number of iterations for the Morgan index computation. When the number of iterations increases, the topological information vehiculated by the Morgan index becomes more and more specific to the graphs. Pairs of vertices having at the same time identical atom type and topological properties are therefore less and less likely to occur. This results in a systematic decrease of the computation time, because the number of nodes of the product graph

automatically decreases, but on the other hand, the similarity between molecules may be difficult to assess if their description becomes too specific. This suggests that the step of the Morgan process that performs the optimum trade off between the uniform and the molecular-specific descriptions of vertices needs to be found.

## 2.3 Preventing totters

A second avenue to modify the original graph kernel is to modify the probability (2.3). 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 walk of the form $w = v_0, \ldots, v_n$ 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 walk 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. Figure 2.4 illustrates this idea.

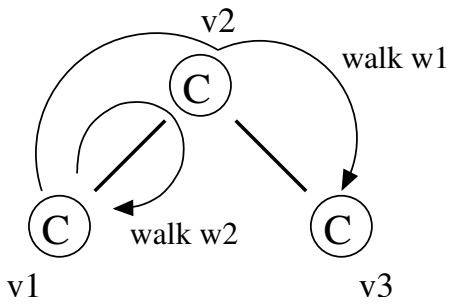


Figure 2.4: Illustration of the process of prevention of the tottering walks on a toy example. Walks $w_1$ and $w_2$ are both labeled as `C-C-C`, but the walk $w_2$ corresponds to a tottering walk.

### 2.3.1 Modification of the random walk

A natural way to carry out this modification is to keep the general kernel definition (2.1) but modify the probability model (2.3) as follows:

$$p((v_0, \ldots, v_n)) = p_s(v_0) p_t(v_1|v_0) \prod_{i=2}^{n} p_t(v_i|v_{i-2}, v_{i-1}), \tag{2.4}$$

where for the graph $G$, $p_i(.)$, $p_t(.|.)$, and $p_t(.|.,.)$ satisfy for any $(t,u,v) \in \mathcal{V}_G^3$:

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

Here we assume that $0 < p_q(v), p_q^{(0)}(v) \leq 1$ for each vertex $v$, $p_a(\cdot|u)$ is a probability on $\mathcal{V}_G$ that is only positive on the neighbors of $u$, and $p_a(\cdot|t,u)$ is a probability on $\mathcal{V}_G$ that is only positive on the neighbors of $u$ different from $t$. 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 2.4), which can not follow excursions of the form $u \to v \to u$. In other words, only walks belonging to

$$\mathcal{W}^{NT}(G) = \{w = (v_0, \ldots, v_n) \in \mathcal{W}(G) : v_i \neq v_{i+2}, i = 0, \ldots, n-2\}, \tag{2.5}$$

can have a positive probability under this model. Given this new random walk model, the function (2.1) is still a valid kernel, but the implementation described in Section 2.1.3 can not be used directly anymore.

### 2.3.2 Computation of the new kernel

While walks have been previously defined as the succession of vertices they are made of, one can see a walk as a starting vertex followed by a succession of connected edges. In such a definition, a pair of connected edges provides information about a triplet of vertices of the walk : the starting vertex of the first edge, the vertex that connects them, and the ending vertex of the second edge. A second-order information about the succession of vertices therefore resumes to a first-order one based on the succession of edges. This suggests it should be possible to deal with second-order "vertex-based" random walks models by means of a first-order ones involving edges of the graphs.

Based on this consideration, we now derive an explicit way to perform the computation of the kernel (2.1) under the model (2.4). To do so, we introduce a graph transformation such that the 2nd-order random walk (2.4) in the original graphs factorizes as a first-order Markov process (2.3) in the transformed ones. More precisely, for a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, let the transformed graph $G' = (\mathcal{V}_{G'}, \mathcal{E}_{G'})$ be defined by

$$\mathcal{V}_{G'} = \mathcal{V}_G \cup \mathcal{E}_G,$$

and

$$\begin{aligned} \mathcal{E}_{G'} = &\big\{\big(v, (v,t)\big) : v \in \mathcal{V}_G, (v,t) \in \mathcal{E}_G\big\} \\ &\cup \big\{\big((u,v),(v,t)\big) : (u,v),(v,t) \in \mathcal{E}_G, u \neq t\big\}. \end{aligned} \tag{2.6}$$

The vertices of the transformed graph $G'$ can therefore correspond either to edges or vertices of the original graph $G$. Among all walks $\mathcal{W}(G')$ on $G'$, let us consider the

subset of walks that start on an arbitrary vertex in $V$, that is the set

$$\mathcal{W}^{\{\mathcal{V}_G\}}(G') = \{w' = (v'_0, \ldots, v'_n) \in \mathcal{W}(G') : v'_0 \in \mathcal{V}_G\}\,. \tag{2.7}$$

Note that from the definition of the transformed graph edges, it is easy to check that any walk $w' = v'_0 \ldots v'_n \in \mathcal{W}(G')$ starting with a vertex $v'_0 \in \mathcal{V}_G$ must subsequently be made of vertices $v'_i,\ i = 1, \ldots, n$ corresponding to edges of $G$. This construction is illustrated in Figure 2.5. We define the labeling function $l'$ of the transformed graph $G'$ as follows :

- for a node $v' \in \mathcal{V}_{G'}$ the label is either $l'(v') = l(v')$ if $v' \in \mathcal{V}_G$, or $l'(v') = l(v)$ if $v' = (u, v) \in \mathcal{E}_G$.
- for an edge $e' = (v'_1, v'_2)$ between two vertices $v'_1 \in \mathcal{V}_G \cup \mathcal{E}_G$ and $v'_2 \in \mathcal{E}_G$, the label is simply given by $l'(e') = l(v'_2)$.

This labeling is also illustrated in Figure 2.5.

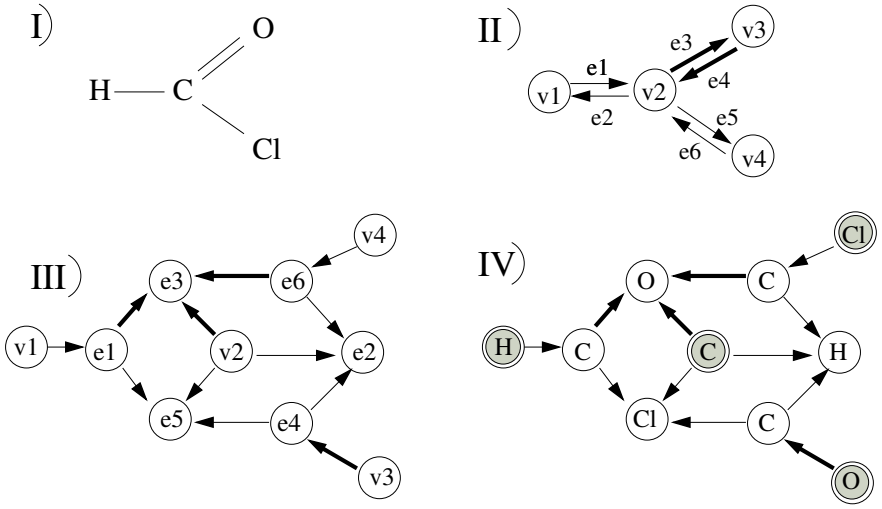


Figure 2.5: The graph transformation. I) The original molecule. II) The corresponding graph $G = (\mathcal{V}_G, \mathcal{E}_G)$. III) The transformed graph. IV) The labels on the transformed graph. Note that different widths stand for different edges labels, and gray nodes are the nodes belonging to $\mathcal{V}_G$.

Let us consider the map $f : \mathcal{W}^{NT}(G) \to \mathcal{V}^*_{G'}$ defined by:

$$f\,(v_0 \ldots v_n) = v'_0 \ldots v'_n,$$

with

$$\begin{cases} v'_0 = v_0 \in \mathcal{V}_G, \\ v'_i = (v_{i-1}, v_i) \in \mathcal{E}_G, \text{ for } i = 1, \ldots, n. \end{cases} \tag{2.8}$$

This definition gives rise to the following proposition, whose proof can be found in Appendix 2.5 :

**Proposition 2.** *$f$ is a bijection between $\mathcal{W}^{NT}(G)$ and $\mathcal{W}^{\{\mathcal{V}_G\}}(G')$ and for any walk $w \in \mathcal{W}^{NT}(G)$ we have $l(w) = l'(f(w))$.*

Finally, let the functional $p' : \mathcal{V}^*_{G'} \to \mathbb{R}$ be derived from (2.4) by:

$$p'\left((v'_0, \ldots, v'_n)\right) = p'_s(v'_0) \prod_{i=1}^{n} p'_t(v'_i|v'_{i-1}), \tag{2.9}$$

with

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

and

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

Note that only walks belonging to $\mathcal{W}^{\{\mathcal{V}_G\}}(G')$ have a positive value under $p'$.

Based on the definitions of $f$ and $p'$, we can state the following result, whose proof is postponed in Appendix 2.5 :

**Theorem 3.** *Under the bijection $f : \mathcal{W}^{NT}(G) \to \mathcal{W}^{\{\mathcal{V}_G\}}(G')$ defined in (2.8), for any walk $w \in \mathcal{W}^{NT}(G)$ we have $p(w) = p'(f(w))$ .*

We have defined in Proposition 2 a graph transformation showing a one to one correspondence between a particular subset of the walks of the transformed graph (the set $\mathcal{W}^{\{\mathcal{V}_G\}}(G')$) and the set of non-tottering walks of the original graph (the set $\mathcal{W}^{NT}(G)$). Moreover we introduced a 1st-order Markov functional (2.9) on the transformed graph, positive only on this particular subset of walks $\mathcal{W}^{\{\mathcal{V}_G\}}(G')$, that corresponds to the 2nd-order probability distribution (2.4) that was previously defined on the original graph to prevent totters. Finally, because our graph transformation preserves the walk labeling, we can immediately deduce the following:

**Corollary 4.** *For any two graphs $G_1$ and $G_2$, the kernel (2.1) based on the second-order Markov random walk model (2.4) can be expressed in terms of the transformed graphs $G'_1$ and $G'_2$ by:*

$$K\left(G_1, G_2\right) = \sum_{\substack{(w'_1, w'_2) \\ \in \mathcal{V}^*_{G'_1} \times \mathcal{V}^*_{G'_2}}} p'_1(w'_1) p'_2(w'_2) K_{\mathcal{L}}\left(l'(w'_1), l'(w'_2)\right),$$

*where $p'_1$ (resp. $p'_2$) is the probability distribution on $\mathcal{V}^*_{G_1}$ (resp. $\mathcal{V}^*_{G_2}$) defined by (2.9).*

This shows that computing the $K(G_1, G_2)$ under the 2nd-order Markov model (2.4) 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 (2.9). This can therefore be carried out using the computation scheme described in Section 2.1.3, at the expense of an increased complexity.

More precisely, consider a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, whose maximum vertex out-degree is $d^+$, and the graph $G' = (\mathcal{V}_{G'}, \mathcal{E}_{G'})$ resulting from its transformation. By definition of $\mathcal{V}_{G'}$, $|\mathcal{V}_{G'}| = |\mathcal{V}_G| + |\mathcal{E}_G|$. Moreover, from the two steps appearing in the definition of $\mathcal{E}_{G'}$, we also have $|\mathcal{E}_{G'}| \leq |\mathcal{E}_G| + (d^+ - 1)|\mathcal{E}_G| = d^+|\mathcal{E}_G|$. Finally, it is easy to check that the node of maximum out-degree in the transformed graph is precisely the node of maximum out-degree in the original graph. This is due to the fact that the nodes of $G'$ corresponding to nodes of $G$ have the same degree that their homologs, and that the nodes in $G'$ corresponding to edges of $G$ have a degree equal to those of the nodes being reached by the edges in $G$ minus one (to prevent tottering). From Section 2.1.3, the complexity of the kernel between two graphs $G_1 = (\mathcal{V}_{G_1}, \mathcal{E}_{G_1})$ and $G_2 = (\mathcal{V}_{G_2}, \mathcal{E}_{G_2})$ writes as $O\left(|\mathcal{E}_{G_1}||\mathcal{E}_{G_2}| \sum_{i=1}^{N-1} \min((d_1^+ d_2^+)^i, |\mathcal{V}_{G_1}||\mathcal{V}_{G_2}|)\right)$. As a result, if we now consider the graphs $G'_1 = (\mathcal{V}_{G'_1}, \mathcal{E}_{G'_1})$ and $G'_2 = (\mathcal{V}_{G'_2}, \mathcal{E}_{G'_2})$ obtained by transforming $G_1$ and $G_2$, this complexity is of order
$O\left(d_1^+ d_2^+ |\mathcal{E}_{G_1}||\mathcal{E}_{G_2}| \sum_{i=1}^{N-1} \min\left((d_1^+ d_2^+)^i, (|\mathcal{V}_{G_1}| + |\mathcal{E}_{G_1}|)(|\mathcal{V}_{G_2}| + |\mathcal{E}_{G_2}|)\right)\right)$.

## 2.4 Experiments

In our experiments, we adopted the parametrization proposed in Kashima et al. (2004). For a given graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, a single parameter $p_q$ is used to define the 1st-order random walk model as follows, for any $u, v \in \mathcal{V}_G$:

- $p_0(v) = 1/|\mathcal{V}_G|$
- $p_q(v) = p_q < 1$
- $p_a(v|u) = \begin{cases} 1/d^+(u) \text{ if } (u, v) \in \mathcal{E}_G \\ 0 \text{ otherwise} \end{cases}$

This way, the emission probability distribution is chosen to be uniform over the set of edges, a constant ending probability is introduced for every node of the graph, and transition probabilities are made uniform over the set of neighbors of the nodes. When we do not have prior knowledge about the data, this seems to be a natural way to parametrize the model.

To filter tottering walks, we adapt this model to define in a similar way the 2nd-order random walk model (2.4) introduced in Section 2.3.1. The main differences between the two models concern the functional $p_q(v)$, $p_q^{(0)}(v)$, and $p_a(v|t, u)$. Indeed, in the first step of the random walk process, the walk is not subject to tottering and we

can consider the same first order transition functional $p_a(v|u)$ and ending probabilities $p_q^{(0)}(v)$. In the following steps however, we may have to set the ending probability $p_q(v)$ to one, in order to explicitly kill random walks when reaching a node with only one neighbor, because in this case, the only possibility to continue the walk is to "totter" to the previous node. The definition of $p_a(v|t,u)$ 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 $v$. This leads to the following 2nd-order Markov model, for any $u, v, w \in \mathcal{V}_G$:

- $p_0(v) = 1/|\mathcal{V}_G|$
- $p_a(v|u) = \begin{cases} 1/d^+(u) \text{ if } (u,v) \in \mathcal{E}_G \\ 0 \text{ otherwise} \end{cases}$
- $p_a(v|t,u) = \begin{cases} 1/(d^+(u)-1) \text{ if } (u,v) \in E \text{ and } v \neq t \\ 0 \text{ otherwise} \end{cases}$
- $p_q^{(0)}(v) = p_q$
- $p_q(v) = \begin{cases} 1 \text{ if } d^+(v) = 1 \\ p_q \text{ otherwise} \end{cases}$

The classification experiments described below were carried out with a support vector machine based on the different kernel tested. Each kernel was implemented in C++, and we used the free and publicly available GIST[3] software to perform SVM classification. No optimization of the parameters required by GIST was carried out. The only option specified was the *-radial* option, which converts the kernel into a radial basis function, a standard way to normalize the data. Two datasets of chemical compounds were used. Both gather results of mutagenicity assays, and while the first one (Debnath et al., 1991) is a standard benchmark for evaluating chemical compounds classification, the second one (Helma et al., 2004) was introduced more recently.

Generally speaking, focusing only on the global accuracy is hazardous to analyze classification results and may lead to wrong conclusions. This is particularly true when the dataset is *unbalanced*, which means that one of the classes is over-represented compared to the other. A safer approach is to describe the classifier using *ROC* analysis (Gribskov and Robinson, 1996), and consider the *sensitivity/specificity* rates. Sensitivity is defined as the ratio between the correctly classified positive data (the *true positives*) and the total number of positive data (the sum of *true positive* and *false negative* data). It therefore accounts for the proportion of positive data that will be retrieved by the classifier. Similarly, specificity accounts for the proportion of negative data that the classifier will correctly find (the ratio between *true negative* data and the

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

whole negative data, that is, the *true negative* plus the *false positive*). Clearly, a good classifier will show a high sensitivity together with a good specificity. Moreover, recall from Section 1.3 that since the SVM prediction is obtained by thresholding a score function (the prediction being +1 if the score is positive, and -1 otherwise), varying the decision threshold makes it possible to draw the evolution of the true positive rate versus the false positive rate in the ROC curve, from which we can derive a global indicator of the performance of the classifier: the *AUC*, Area Under the (ROC) Curve. The AUC of an ideal classifier would be 1 (the positive data would be the first to be recognized as positive according to their scores), while for a random classifier it would be 0.5 (Fawcett, 2003).

### 2.4.1 First dataset

This dataset contains 230 chemical compounds (aromatic and hetero-aromatic nitro compounds) tested for mutagenicity on *Salmonella typhimurium*. A SAR analysis on this dataset was first conducted by Debnath et al. (1991), who identified two subsets of the data: 188 compounds considered to be amenable to regression and 42 compounds that could not easily be fitted by regression. In this study we mainly focus on the first set of 188 compounds. These compounds can be split into two classes: 125 positive examples with high mutagenic activity (positive levels of log mutagenicity), and 63 negative examples with no or low mutagenic activity. Each chemical compound is represented as a graph with atoms as vertices and covalent bonds as edges. This subset of 188 compounds was already used in the original paper Kashima et al. (2004), and in a similar way, kernels are evaluated here by their leave-one-out error. AUC will be our quality criterion.

Table 2.1 shows the results we can get with the kernel as it is formulated in Kashima et al. (2004). They will be our reference results to evaluate the impact of the proposed extensions. This table shows a consistent increase in the AUC when the parameter $p_q$ decreases, that is to say when the kernel favors long walks.

Figure 2.6 shows the effect of removing the tottering walks with the original kernel formulation for distinct values of $p_q$. The curve reveals that the relationship between small $p_q$ and high AUC observed with the original formulation of the kernel does not rigorously hold any longer when tottering walks are filtered. Indeed, we can find in this case an optimum value of $p_q$ around 0.1. Above this value, we can notice on the one hand a small but consistent increase on classification when tottering walks are removed, and on the other hand that the effect of this extension becomes smaller when $p_q$ increases. This is probably due to the fact that when $p_q$ increases, the walks taken into account by the kernel tend to become shorter and are therefore less likely to totter. When $p_q \leq 0.1$, the AUC decreases and becomes smaller than the one obtained with the original formulation for $p_q = 0.001$.

Table 2.2 shows the AUC results for distinct values of $p_q$ combined with the introduction of Morgan indices. It reveals that the introduction of Morgan indices can

| $p_q$ | Accuracy | Sensitivity | Specificity | AUC |
|---|---|---|---|---|
| 0.01 | 89.4 | 88.8 | 90.4 | 94.4 |
| 0.05 | 89.4 | 88.8 | 90.4 | 94.2 |
| 0.1 | 89.9 | 89.6 | 90.4 | 94.0 |
| 0.2 | 90.4 | 90.4 | 90.4 | 93.8 |
| 0.3 | 90.4 | 90.4 | 90.4 | 93.6 |
| 0.4 | 88.8 | 88.0 | 90.4 | 93.4 |
| 0.5 | 88.8 | 88.0 | 90.4 | 93.0 |
| 0.6 | 88.3 | 87.2 | 90.4 | 92.7 |
| 0.7 | 87.2 | 85.6 | 90.4 | 92.2 |
| 0.8 | 86.7 | 84.8 | 90.4 | 91.2 |
| 0.9 | 83.5 | 82.4 | 85.7 | 89.2 |

Table 2.1: Classification of the 1st dataset, with the original formulation of the kernel function for different values of the parameter $p_q$.

always increase the classification results, and interestingly, the optimal index to be used depends on the value of $p_q$ : it is generally smaller for little values of $p_q$. This reflects the fact that we have to add more specificity in the atoms labels for large $p_q$, since only walks involving a few atoms will be taken into account. However, no prior rule can define a precise relation between the Morgan index and the parameter $p_q$.

| $p_q$ | 0.01 | 0.05 | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|---|---|
| MI = 0 | 94.4 | 94.2 | 94.0 | 93.6 | 93.0 | 92.2 | 89.2 |
| MI = 1 | 94.4 | 94.2 | 93.8 | 93.2 | 92.7 | 92.2 | 92.0 |
| MI = 2 | **96.1** | **96.0** | **95.9** | **95.2** | 94.3 | 93.6 | 93.1 |
| MI = 3 | 94.6 | 94.7 | 94.7 | 94.9 | **94.8** | **94.8** | **94.6** |
| MI = 4 | 93.3 | 93.3 | 93.2 | 93.3 | 93.1 | 93.0 | 92.6 |
| MI = 5 | 92.3 | 92.4 | 92.5 | 92.8 | 93.2 | 93.4 | 93.5 |
| MI = 6 | 91.6 | 91.8 | 92.0 | 92.6 | 92.8 | 92.9 | 92.8 |
| MI = 7 | 90.2 | 90.1 | 90.1 | 90.1 | 90.1 | 90.1 | 90.2 |
| MI = 8 | 86.9 | 87.1 | 87.3 | 87.7 | 88.1 | 88.3 | 88.4 |
| MI = 9 | 80.5 | 80.8 | 81.5 | 81.6 | 81.7 | 81.9 | 81.7 |
| MI = 10 | 72.8 | 72.8 | 73.7 | 76.2 | 77.1 | 77.6 | 77.9 |

Table 2.2: AUC for the 10 first Morgan indices and different ending probabilities, 1st dataset.

Table 2.3 shows the AUC results when tottering walks have been filtered. The classification results show the same behavior of the kernel with respect to $p_q$ and the Morgan indices when the tottering walks have been filtered. The values obtained are

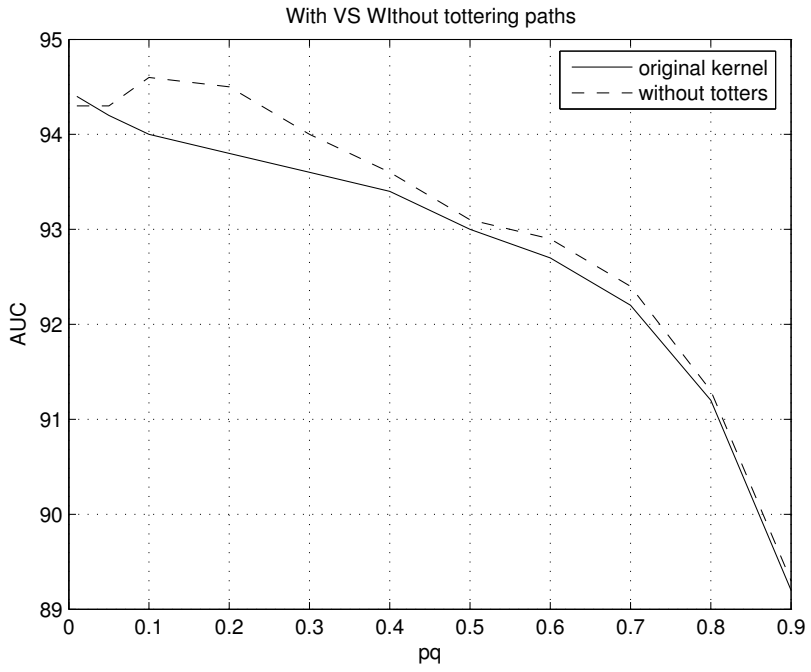


Figure 2.6: AUC for distinct values of $p_q$, with and without filtering the tottering walks, 1st dataset.

actually sensibly equal, which suggests that filtering the tottering walks provides little additional information. We can however notice that performances are globally reduced when $p_q$ becomes smaller. Tottering between atoms made specific to the molecule therefore accounts for graphs similarity when long walks are taken into account.

Finally, results about computation times are presented in Figure 2.7. The top curve plots the evolution of the time needed to compute the kernels when different Morgan indices were introduced. More precisely it plots the ratio between the time needed for a given iteration of the Morgan process, and the time initially required. Note that the $y$-axis is in log-scale, so that we can notice a drastic decrease in the computational cost. For example, at the 3rd iteration of the process, computation time is reduced by a factor around 40 when tottering walks have not been filtered. Remind that when Morgan indices are introduced, atoms are made more specific to the molecule they belong, and as a consequence fewer atoms are apariated in the product graph, which makes the matrix inversion cheaper. This effect is even stronger when filtering the totters. The bottom curve presents the impact of totter removal on the computation times. The curve shows ratio between the computation times of the original formulation and the totters filtering. This ratio becomes smaller with the Morgan process, but while the computation without totters was initially more than a hundred time longer than with totters, it remains at least ten times longer with high

| $p_q$ | 0.01 | 0.05 | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|---|---|
| MI = 0 | 94.3 | 94.3 | 94.6 | 94.0 | 93.1 | 92.4 | 89.3 |
| MI = 1 | 94.0 | **95.1** | 94.2 | 94.0 | 93.2 | 92.3 | 92.0 |
| MI = 2 | **94.9** | 94.9 | **95.2** | **95.4** | 94.5 | 93.6 | 93.1 |
| MI = 3 | 93.3 | 93.4 | 93.6 | 94.7 | **94.8** | **94.8** | **94.6** |
| MI = 4 | 92.5 | 92.4 | 92.7 | 93.3 | 93.2 | 92.9 | 92.6 |
| MI = 5 | 90.5 | 90.7 | 91.0 | 92.6 | 93.1 | 93.4 | 93.4 |
| MI = 6 | 88.2 | 88.5 | 89.9 | 92.1 | 92.8 | 92.9 | 92.8 |
| MI = 7 | 84.7 | 86.4 | 88.5 | 90.2 | 90.1 | 90.2 | 90.2 |
| MI = 8 | 69.2 | 73.9 | 81.4 | 87.4 | 88.0 | 88.2 | 88.3 |
| MI = 9 | 57.1 | 60.6 | 70.7 | 81.3 | 81.8 | 81.7 | 81.7 |
| MI = 10 | 49.2 | 48.8 | 52.7 | 73.8 | 76.8 | 78.2 | 78.3 |

Table 2.3: AUC for the 10 first Morgan indices and different ending probabilities, when tottering walks have been filtered, 1st dataset.

Morgan indices.

As a comparison, Table 2.4 gathers 10-fold cross-validated accuracy results already obtained for the classification of this set of 188 compounds. These methods can be split in three categories : those relying on global molecular properties[4] (*lin.Reg, NN, DT*), those considering the structure of the molecules as a set of atoms and connecting bonds (*Progol1*), and those involving the two representations (*Progol2, Sebag, Kramer*). The best 10-fold cross-validated accuracy corresponding to our previous experiments is 91.2%. As we can notice from Table 2.4, this result is better than those based on one of the two molecular representations, but it is below those obtained by methods that combine both representations. This table reveals that there is a significant gap between the *Progol1* and *Progol2* results, which are obtained using the same algorithm when the global descriptors are considered or not as an additional source of information. This suggests that the information contained in the two descriptions may be complementary. Moreover, the best result reported Kramer and De Raedt (2001) deals with the structure of the molecule via a fragment-based characterization, which, as we already mentioned, shows some similarities with the graph kernel approach. It seems therefore reasonable to draw the hypothesis that the results of the graph kernel approach may be improved if such a combination of information about the molecules is used.

Little work has been carried out on the 42 compounds that constitute the "non regression friendly part" of the dataset. To our knowledge, the only results were published in King et al. (1996), and are summarized in Table 2.5. The fundamental

[4]for instance, the molecular hydrophobicity (logP), the energy of the lowest unoccupied molecular orbital (LUMO) and two additional binary descriptors coding for the presence of particular features in the molecule.

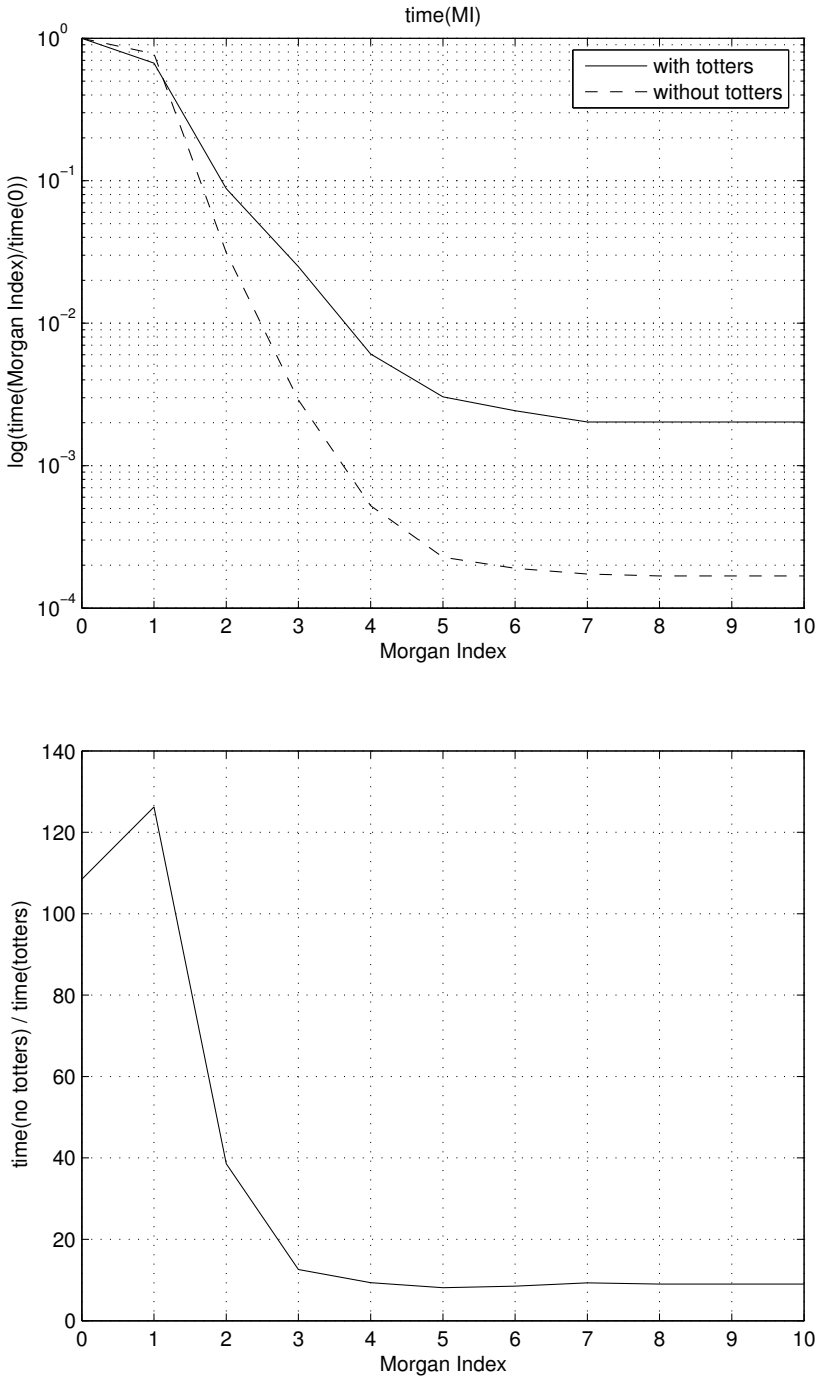


Figure 2.7: Top : Time needed to compute the kernel for the 10 first iterations of the Morgan process. Bottom : Ratio between computation times with or without filtering the totters for the 10 first iterations of the Morgan Process, 1st dataset.

| Lin.Reg | DT | NN | Progol1 | Progol2 | Sebag | Kramer | Graph kernels |
|---|---|---|---|---|---|---|---|
| 89.3% | 88.3% | 89.4% | 81.4% | 87.8% | 93.3% | 95.7% | 91.2% |

Table 2.4: Accuracy results obtained for the 10-fold Cross-Validation of the Mutag dataset. *Lin.Reg* (Linear Regression), *DT* (Decision Tree), *NN* (Neural Network) and *Progol1/2* (Inductive Logic Programming) : King et al. (1996) ; *Sebag* : Sebag and Rouveirol (1997) ; *Kramer* : Kramer and De Raedt (2001).

difference with the "friendly part" of the dataset, lies in the fact that here, the best result was obtained using only the 2D structure of the compounds (with the *Progol1* method). Using our graph kernels, we can reach 88.1% of correct classification using a similar leave-one-out procedure. Outperforming all the results from Table 2.5, this result shows that the graph kernel approach is indeed efficient when the relevant information is to be sought in the structure of the molecules.

| Lin.Reg | Lin.Reg+ | DT | ANN | Progol1 | Progol2 | Graph Kernels |
|---|---|---|---|---|---|---|
| 66.7% | 71.8% | 83.3% | 69.0% | 85.7% | 83.3% | 88.1% |

Table 2.5: Accuracy results obtained for the leave-one-out classification of the "unfriendly part" of the Mutag dataset. *Lin.Reg* (Linear Regression), *DT* (Decision Tree), *NN* (Neural Network) and *Progol1/2* (Inductive Logic Programming) : King et al. (1996).

Independently of our work, related graph kernels for chemoinformatics applications were recently introduced (Ralaivola et al., 2005) . Their formulation is driven by the usual molecular fingerprinting process, and several kernel functions are proposed based on variations of the Tanimoto coefficient. Different ways of fingerprinting the molecules are considered, and in particular, some experiments compare standard hashed fingerprints (such as *Daylight* fingerprints) with exhaustive fingerprints, for molecular fragments up to a given length (which was set to 10). In exhaustive fingerprints, a dimension is introduced for every possible molecular fragment, which is closely linked to the description of molecules related to the graph kernels introduced here. Although the performances of these different configurations are similar, this study tends to reveal that the hashing process leads to a decrease in the classification accuracy. More precisely, the best result for exhaustive fingerprints reaches 87.8% of correct leave-one-out classification, while it is 87.2% when the fingerprints are hashed. Using our graph kernels, we can reach a leave-one-out accuracy of 91%, which indicates that the marginalized graph kernels approach may compare favorably to classical hashed-fingerprints. Note however that results in Ralaivola et al. (2005) were not obtained using SVM, but using the Voted Perceptron algorithm, an algorithm known to provide comparable results, and that further refinements of their kernels lead to an optimal accuracy of 91.5%.

### 2.4.2 Second dataset

The second database considered was recently introduced in Helma et al. (2004). It consists of 684 compounds classified as mutagens or non-mutagens according to a test known as the *Salmonella*/microsome assay. The classes are well balanced with 341 mutagens compounds for 343 non-mutagens ones. Although the biological property to be predicted is the same as the one of the previous section, the two datasets are fundamentally different. While Debnath et al. (1991) focused on a particular family of molecules (aromatic and hetero-aromatic nitro compounds), this dataset involves a set of very diverse chemical compounds, qualified as *noncongeneric* in the original paper. To predict mutagenicity, the model therefore needs to solve different tasks : in the first case it has to detect subtle differences between homogeneous structures, while in the second case it must seek for regular patterns within a set of structurally different molecules. As stated in Helma et al. (2004), toxicity is a very complex and multi-factor mechanism, so that diverse datasets need to be considered in order to be able to predict mutagenicity in real-world applications. Finally, note that this dataset is public and a further description can be found in Helma et al. (2004).

We applied different graph kernels to this dataset in order to compare our approach to the results presented in Helma et al. (2004). Several machine learning algorithms have been used in that paper (namely SVM, decision trees and rule learning), based on a molecular-fragment characterization of molecules. In their method, a set of substructures occurring frequently in mutagenic compounds, but seldomly in non-mutagens ones is defined, and molecules are represented by bit-strings indicating the presence or absence of these substructures. Tables 2.6 and 2.7 gather results on this dataset using the original and totters-filtering versions of the kernel, for several values of $p_q$ and different iterations of the Morgan process. Following Helma et al. (2004), we performed classifications by a 10-fold cross-validation procedure, and performances are evaluated according to the *accuracy*, *sensitivity* and *specificity* values of the models.

A quick inspection of these two tables reveals that, similarly to the original paper, the test sensitivity and specificity rates are always similar. This means that the different models obtained can be used to predict either mutagenicity or non-mutagenicity, with a similar degree of confidence. From this consideration, we base our analysis of the results on the global test accuracy of the models.

Several conclusions can be drawn from these tables. First, when no Morgan indices are introduced (i.e. $MI = 0$), we can note from both tables that test accuracy systematically increases when the parameter $p_q$ decreases. This is consistent with the experiments carried out with the previous dataset and suggests that it is worth considering long walks. Moreover, when we compare the two tables, we note that filtering the totters systematically enhances the classification, which comforts the intuition that this kind of walks adds noise to the description of the molecules.

Concerning the introduction of the Morgan indices, we can note from the two tables that, for any value of $p_q$ considered, classification is improved for the first iteration of

| MI | $p_q$ | Training Accuracy | Test Accuracy | Test Sensitivity | Test Specificity |
|---|---|---|---|---|---|
| 0 | 0.1 | 94.47 | 76.02 | 73.34 | 78.67 |
| | 0.2 | 93.85 | 75.14 | 72.65 | 77.77 |
| | 0.3 | 92.65 | 74.20 | 71.14 | 77.26 |
| | 0.4 | 92.06 | 73.36 | 70.76 | 75.90 |
| | 0.5 | 92.77 | 73.32 | 71.61 | 75.03 |
| 1 | 0.1 | 96.03 | 79.01 | 77.06 | 80.97 |
| | 0.2 | 96.00 | 79.32 | 77.55 | 81.08 |
| | 0.3 | 96.08 | 78.70 | 77.65 | 79.73 |
| | 0.4 | 96.11 | 78.40 | 76.45 | 80.30 |
| | 0.5 | 96.01 | 77.17 | 76.00 | 78.20 |
| 2 | 0.1 | 97.20 | 78.09 | 76.95 | 79.25 |
| | 0.2 | 97.13 | 77.89 | 77.33 | 78.47 |
| | 0.3 | 97.02 | 78.70 | 78.43 | 78.93 |
| | 0.4 | 96.91 | 78.34 | 78.11 | 78.56 |
| | 0.5 | 96.82 | 78.20 | 77.71 | 78.70 |
| 3 | 0.1 | 97.42 | 75.32 | 73.62 | 76.95 |
| | 0.2 | 97.14 | 75.00 | 73.31 | 76.67 |
| | 0.3 | 96.92 | 75.88 | 74.55 | 77.21 |
| | 0.4 | 96.71 | 75.00 | 74.04 | 76.12 |
| | 0.5 | 96.57 | 75.27 | 73.81 | 76.69 |
| 4 | 0.1 | 98.03 | 71.45 | 68.53 | 74.41 |
| | 0.2 | 97.55 | 71.80 | 69.47 | 74.20 |
| | 0.3 | 97.29 | 72.10 | 69.77 | 74.50 |
| | 0.4 | 97.10 | 71.52 | 69.79 | 73.21 |
| | 0.5 | 96.87 | 72.18 | 69.86 | 74.53 |
| 5 | 0.1 | 97.65 | 66.76 | 64.32 | 69.30 |
| | 0.2 | 97.24 | 67.73 | 65.56 | 69.80 |
| | 0.3 | 96.94 | 66.99 | 64.65 | 69.27 |
| | 0.4 | 96.58 | 66.85 | 65.11 | 68.72 |
| | 0.5 | 96.38 | 66.73 | 64.62 | 68.86 |

Table 2.6: Classification results for the 2nd mutagenicity dataset, with the tottering walks.

| MI | $p_q$ | Training Accuracy | Test Accuracy | Test Sensitivity | Test Specificity |
|---|---|---|---|---|---|
| 0 | 0.1 | 95.60 | 77.37 | 76.59 | 78.15 |
| | 0.2 | 95.11 | 75.46 | 72.38 | 78.63 |
| | 0.3 | 94.48 | 74.93 | 71.95 | 77.82 |
| | 0.4 | 93.65 | 75.13 | 71.81 | 78.44 |
| | 0.5 | 93.22 | 74.18 | 71.58 | 76.77 |
| 1 | 0.1 | 97.22 | 77.87 | 74.90 | 80.90 |
| | 0.2 | 96.43 | 78.89 | 76.65 | 81.16 |
| | 0.3 | 96.21 | 79.06 | 77.28 | 81.00 |
| | 0.4 | 96.20 | 78.54 | 77.25 | 79.84 |
| | 0.5 | 96.22 | 78.47 | 76.65 | 80.03 |
| 2 | 0.1 | 97.72 | 76.59 | 72.03 | 81.21 |
| | 0.2 | 97.52 | 77.95 | 75.35 | 80.47 |
| | 0.3 | 97.20 | 78.04 | 77.01 | 79.08 |
| | 0.4 | 97.04 | 78.28 | 77.77 | 78.80 |
| | 0.5 | 96.90 | 78.03 | 77.54 | 78.53 |
| 3 | 0.1 | 97.97 | 75.18 | 69.82 | 80.54 |
| | 0.2 | 97.79 | 75.87 | 72.99 | 78.69 |
| | 0.3 | 97.53 | 75.48 | 74.36 | 76.63 |
| | 0.4 | 97.06 | 75.54 | 73.98 | 77.10 |
| | 0.5 | 96.78 | 75.62 | 75.22 | 76.00 |

Table 2.7: Classification results for the 2nd mutagenicity dataset, when tottering walks were removed.

the process, after what it systematically decreases. This means that although the first step of the Morgan process could improve the expressive power of the kernel, the information introduced into the description of the molecule becomes too specific from the second iteration. Interestingly, we can also notice that for a given index of the Morgan process, the optimal value of $p_q$ is not the smallest one any longer.

Filtering the totters after the introduction of the Morgan indices have a somehow ambiguous effect. It does not show a consistent trend with respect to the parameter $p_q$. However, the two optimal results show a 79.32% and 79.06% test accuracy, so that results are globally similar.

Finally, note that the computation times needed to compute the different kernels follow the same behavior as the results presented in the previous subsection.

Helma et al. (2004) pointed out the need to consider structurally diverse datasets such as this one in order to be able to model multi factor mechanisms such as toxicity. Although the classification accuracy provides a general measure of the effectiveness of the algorithm, it is of limited help to quantify its ability to handle the diversity of the dataset. For instance, a situation where the subset of correctly classified data shows a smaller diversity compared to the global dataset actually makes sense. This situation means that the algorithm is only efficient in a particular subspace of the chemical space defined by the whole dataset, which is actually likely to occur, and reveals that the method fails to handle the diversity of the dataset. Analyzing the diversity of the classification results is therefore useful to give fair conclusions about the method. Table 2.8 shows the values of a diversity criterion measured on the whole dataset and on several subsets: the subsets of positive compounds, negative compounds, correctly classified compounds, positive compounds that were correctly classified and negative compounds that where correctly classified. These values were computed for two kernels that correspond to optimal results in the previous tables : those obtained using the tottering walks, the first iteration of the Morgan process, and values of $p_q$ of 0.1 and 0.2 [5].

To evaluate the diversity of a subset, we use the average distance of the points of this subset to their center of mass, in the feature-space associated with the kernel (Shawe-Taylor and Cristianini, 2004). Recall from Section 1.3.2 that a kernel function $k$ corresponds to a dot-product between the data mapped to a vector space $\mathcal{F}$ (the so-called *feature-space*): $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. A distance in the feature-space is therefore implicitly defined by the kernel function : $d_{\mathcal{F}}(x_1, x_2) = ||\phi(x_1) - \phi(x_2)||^2 = k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2)$. Our diversity criterion, for the subset $S$ (of cardinality $n_S$), therefore writes as

$$D(S) = \frac{1}{n_S} \sum_{i \in S} d_{\mathcal{F}}(x_i, M),$$

[5] The kernels used actually correspond to the kernels used for the classification, i.e. the kernels obtained after applying the *-radial* option of the *GIST* software to the original kernels.

where $M$ is the center of mass of $S$, that is $M = \frac{1}{n_S} \sum_{i \in S} \phi(x_i)$, which leads to

$$D(S) = \frac{1}{n_S} \sum_{i \in S} k(x_i, x_i) - \frac{1}{n_S^2} \sum_{i,j \in S} k(x_i, x_j).$$

Table 2.8 reveals that the diversities of these different subsets are very similar. Two main conclusions can be drawn from this observation. First, the fact the the diversity of whole dataset equals that of the positive and negative subsets reveals that the supports of these subsets largely overlap and indicates that the classification problem is not trivial. Indeed, in a particularly simple configuration the diversities of the positive and negative subsets would be significantly smaller than that of the dataset as a whole: the two classes of compounds would be clearly separated in the chemical space. Second, the fact that all these values are similar shows that this algorithm is able to correctly classify data regardless of the class they belong to, nor their location in the chemical space. This accounts for the fact that the method is indeed able to handle non-congeneric datasets.

| | All | Pos. | Neg. | Correct | Correct & Pos. | Correct & Neg. |
|---|---|---|---|---|---|---|
| $p_q = 0.1$ | 0.862 | 0.849 | 0.864 | 0.854 | 0.826 | 0.860 |
| $p_q = 0.2$ | 0.868 | 0.856 | 0.870 | 0.860 | 0.834 | 0.864 |

Table 2.8: Diversity values for different subsets of the data, computed from the kernels obtained using the tottering walks, the first iteration of the Morgan process, and two different values of $p_q$. *All* : whole dataset ; *Pos.* : positive compounds ; *Neg.* : negative compounds ; *Correct* : correctly classified compounds ; *Correct & Pos.* : correctly classified positive compounds ; *Correct & Neg.* : correctly classified negative compounds.

Finally, we can note that Tables 2.6 and 2.7 compare quite favorably to the results presented in Helma et al. (2004). Many configurations have been tested in Helma et al. (2004), and the best model reported has an accuracy of 78.5%. With an optimal accuracy of 76%, the original graph kernel with SVM shows a slightly smaller performance, but the extensions introduced here could raise this figure up to more than 79%. Based on our pair of extensions, we have therefore been able to propose models with state-of-the-art performance, and even models performing slightly better. We can however note a slight difference between the two family of models: models from Helma et al. (2004) tend to have a higher sensitivity, while ours show a better specificity. This means that the models from Helma et al. (2004) will correctly classify a larger fraction of the positive data, but this comes at the expense of a larger false positive rate, whereas our models may miss sensibly more true positive data, but the confidence in positive predictions is higher.

## 2.5 Discussion and conclusion

Based on a recently introduced family of graph kernels, we validated in this paper the graph kernel approach for SAR analysis. Experiments revealed in a consistent way that optimal results are obtained when long walks are considered, and this insight is worth to solve the problem of model parametrization. We introduced two extensions to the general formulation, and showed they can actually improve the SAR models in terms of accuracy of the predictions and/or computation times. These two extensions are formulated as pre-processing steps of the algorithm and are therefore completely modular. Moreover, they are based on general graph considerations, and we believe they can be useful in other problems.

The fundamental difference between this approach and other SAR algorithms lies in the fact that the step of feature selection inherent to all other methods is avoided here. In this sense, these kernels provide a kind of universal way to compare molecules. Together with the panel of kernel methods algorithms, this family of graph kernels could be used straightaway to solve different SAR problems, such as clustering or regression tasks for instance, which otherwise typically involve multiple feature selection tasks. On top of that, since it deals with every molecular fragment of the molecules, this model can benefit from structural patterns responsible for activity that have not been discovered yet, and are therefore not included in the set of traditional descriptors. This property however comes at the expense of the interpretability of the model, which has a great interest in medicinal chemistry. Indeed, an interpretable model can give clues to explain the causes of activity or non-activity, and therefore provide chemists with a worthy feedback to carry out molecular optimization in a rational way. An important challenge to the application of these graph kernels in chemoinformatics is to be able to extract information from their infinite dimensional feature-space, and to formalize it in terms of chemical knowledge.

## Acknowledgments

This work was supported by a French-Japanese *SAKURA* grant. Jean-Luc Perret is partly supported by a grant from the *Swiss National Science Foundation*.

## Appendix

### Proof of Proposition 2

For any walk $w = (v_0, \ldots, v_n) \in \mathcal{W}^{NT}(G)$, let $f(w) = (v'_0, \ldots, v'_n)$ defined by (2.8). By definition (2.6), $(v'_0, v'_1) = (v_0, (v_0, v_1)) \in \mathcal{E}_{G'}$, and $\left(v'_i, v'_{i+1}\right) = ((v_{i-1}, v_i), (v_i, v_{i+1})) \in \mathcal{E}_{G'}$ because $v_{i+1} \neq v_{i-1}$ for $i > 0$. Hence $f(w)$ is a walk in $G'$. Moreover $v'_0 \in \mathcal{V}_G$ and $v'_i \in \mathcal{E}_G$ by (2.8), hence $f(w) \in \mathcal{W}^{\{\mathcal{V}_G\}}(G')$ by (2.7).
Conversely, for any $w' = (v'_0, \ldots, v'_n) \in \mathcal{W}^{\{\mathcal{V}_G\}}(G')$, we have $v'_0 = v_0 \in \mathcal{V}_G$ and by easy

induction using the definition of edges (2.6), $v'_i = (v_{i-1}, v_i) \in \mathcal{E}_G$ with $v_{i-1} \neq v_{i+1}$. Hence $w' = f(w)$ with $w = (v_0, \ldots, v_n) \in \mathcal{W}^{NT}(G)$, therefore $f$ is surjective. By definition of $f$ (2.8), it is also clear that $f(w) = f(w') \Rightarrow w = w'$. $f$ is therefore a bijection from $\mathcal{W}^{NT}(G)$ onto $\mathcal{W}^{\{\mathcal{V}_G\}}(G')$.

Moreover, by definition of the labeling $l'$ on $G'$, we obtain for any $w = (v_0, \ldots, v_n) \in \mathcal{W}^{NT}(G)$:

$$\begin{aligned} l'(f(w)) &= l'\left(v_0, (v_0, v_1), \ldots, (v_{n-1}, v_n)\right) \\ &= l(v_0)l(v_1)\ldots l(v_n) \\ &= l(w). \end{aligned}$$

■

## Proof of Theorem 3

From the definition of $p'$, for any $w = (v_0, \ldots, v_n) \in \mathcal{W}^{NT}(G)$, we obtain :

$$\begin{aligned} p'(f(w)) &= p'\left(v_0, (v_0, v_1), \ldots, (v_{n-1}, v_n)\right) \\ &= p'_s(v_0)p'_t\left((v_0, v_1)|v_0\right)\prod_{i=2}^{n} p'_t\left((v_{i-1}, v_i)|(v_{i-2}, v_{i-1})\right) \\ &= p_s(v_0)p_t(v_1|v_0)\prod_{i=2}^{n} p_t\left(v_i|v_{i-2}, v_{i-1}\right) \\ &= p(w). \end{aligned}$$

■

Chapter 3

# Tree-pattern graph kernels

The pioneer graph kernels introduced in the previous chapter show how to map graphs to an infinite-dimensional feature space indexed by linear subgraphs, and compute an inner product in that space. The resulting graph kernels compare two graphs through their common walks, weighted by a function of their lengths (Gärtner et al., 2003) or by their probability under a random walk model on the graphs (Kashima et al., 2004). While this representation might appear restrictive, these kernels led to promising empirical results, often comparing to state-of-the-art approaches in the fields of chemoinformatics (Mahé et al., 2005; Ralaivola et al., 2005) and bioinformatics (Borgwardt et al., 2005; Karklin et al., 2005).

Nevertheless, Ramon and Gärtner (2003) highlighted the limited expressiveness of graph kernels based on linear features, showing in particular that many different graphs can be mapped to the same point in the corresponding feature space. Figure 3.1 illustrates this issue on a simple example. On the other hand, they also showed that computing a perfect graph kernel, that is, a kernel mapping non-isomorphic graphs to distinct points in the feature space, is NP-hard. This suggests that the expressiveness of graph kernels must be traded for their computational complexity. As a first step towards a refinement of the feature space used in walk-based graph kernels, Ramon and Gärtner (2003) introduced a kernel function comparing graphs on the basis of their common subtrees. This representation looks promising in particular in chemoinformatics, because physicochemical properties of atoms are known to be related to their topological environment that could be well captured by subtrees. However, the relationship between the new subtree-based kernel and previous walk-based kernels was not analyzed in details, and the relevance of the new kernel was not tested empirically.

Our motivation in this chapter is to study in detail, both theoretically and empirically, the relevance of subtree features for graph kernels, and in particular to assess the benefits they bring compared to walk-based graph kernels. For that purpose we first revisit the formulation introduced by Ramon and Gärtner (2003) and propose two new kernels with an explicit description of their feature spaces and corresponding inner products. We introduce a parameter in the formulations that allows to gradu-

ally increase the complexity of the subtrees used as features to represent the graphs, the notion of complexity depending on the formulation. By decreasing the parameter we recover classical walk-based kernels, and by increasing it, we can empirically observe in detail the effect of increasing the number and the complexity of the tree features used to represent the graphs. Both formulations can be efficiently computed by dynamic programming, in the spirit of the kernel proposed by Ramon and Gärtner (2003). When the size of allowed subtrees is increased, however, we observe that the practical use of this kernel is limited by the explosion in the number of subtrees occurring in the graphs. In a second step, we therefore introduce two extensions to the initial formulation of the kernels that allow, on the one hand, to extend and generalize their associated feature space, and on the other hand, to remove noisy features that correspond to unwanted subtrees. The different kernels are compared experimentally on two binary classification tasks consisting in discriminating toxic from non-toxic molecules with a SVM.

The remaining of the chapter is organized as follows. Notations and definitions related to graphs and trees are introduced in Section 3.1, followed in Section 3.2 by the definition of a general class of kernels based on the detection of common subtrees. The next section (Section 3.3) revisits the framework introduced in Ramon and Gärtner (2003), from which two particular graph kernels are derived and further extended in Section 3.4. After a discussion about their implementation in Section 3.5, the kernels are validated experimentally (Section 3.6), and we give concluding remarks in Section 3.7.

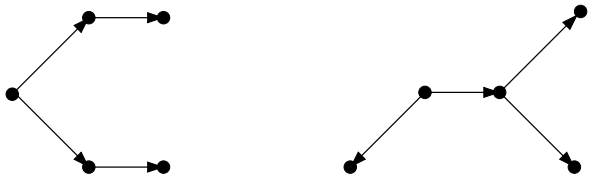

Figure 3.1: Two graphs having the same walk content, namely $\bullet$ : $\times 5$ ; $\bullet\!\!\rightarrow\!\!\bullet$ : $\times 4$ and $\bullet\!\!\rightarrow\!\!\bullet\!\!\rightarrow\!\!\bullet$ : $\times 2$, and consequently mapped to the same point of the feature space corresponding a kernel based on the count of walks (Gärtner et al., 2003).

## 3.1 Notations and definitions

In this section we introduce notations and general definitions related to graphs and trees.

### 3.1.1 Labeled directed graphs

A *labeled graph* $G = (\mathcal{V}_G, \mathcal{E}_G)$ is defined by a finite set of *vertices* $\mathcal{V}_G$, a set of *edges* $\mathcal{E}_G \subset \mathcal{V}_G \times \mathcal{V}_G$, and a labeling function $l : \mathcal{V}_G \cup \mathcal{E}_G \rightarrow \mathcal{A}$ which assigns a *label* $l(x)$ taken

from an alphabet $\mathcal{A}$ to any vertex or edge $x$. We let $|\mathcal{V}_G|$ be the number of vertices of $G$, $|\mathcal{E}_G|$ be its number of edges, and we assume below that a set of labels $\mathcal{A}$ common to all graphs has been fixed. In *directed* graphs, edges are oriented and to each vertex $u \in \mathcal{V}_G$ corresponds a set of *incoming neighbors* $\delta^-(u) = \{v \in \mathcal{V}_G : (v,u) \in \mathcal{E}_G\}$ and *outgoing neighbors* $\delta^+(u) = \{v \in \mathcal{V}_G : (u,v) \in \mathcal{E}_G\}$. We let $d^-(u) = |\delta^-(u)|$ be the *in-degree* of the vertex $u$, and $d^+(u) = |\delta^+(u)|$ be its *out-degree*. A *walk* of length $n$ in the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ is a succession of $n+1$ vertices $(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1}$, such that $(v_i, v_{i+1}) \in \mathcal{E}_G$ for $i = 0, \ldots, n-1$. A *path* is a walk $(v_0, \ldots, v_n)$ with the additional condition that $i \neq j \iff v_i \neq v_j$. Finally, a graph is said to be *connected* if there is a walk between any pair of vertices when the orientation of edges is dropped.

For applications in chemistry considered below, we associate a labeled directed graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ to the planar structure of a molecule. To do so, we let the set of vertices $\mathcal{V}_G$ correspond to the set of atoms of the molecule, the set of edges $\mathcal{E}_G$ to its covalent bonds, and label these graph elements according to an alphabet $\mathcal{A}$ consisting of the different types of atoms and bonds. Note that since graphs are directed, a pair of edges of opposite direction is introduced for each covalent bond of the molecule. Figure 3.2 shows a chemical compound seen as a labeled directed graph.

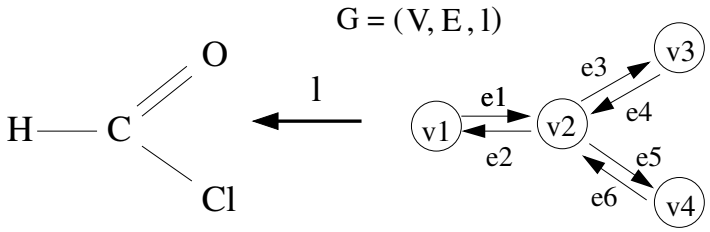


Figure 3.2: A chemical compound seen as a labeled graph

### 3.1.2 Trees

A *tree* $t$ is a directed connected acyclic graph in which all vertices have in-degree one, except one that has in-degree zero. The node with in-degree zero is known as the *root* $r(t)$ of the tree. Nodes with out-degree zero are known as *leaf* nodes, others are called *internal* nodes. Trees are naturally oriented, edges being directed from the root to the leaves. The outgoing neighbors of an internal node are known as its *children*, and the unique incoming neighbor of a node (apart from the root) is known as its *parent*. If two nodes have the same parent, their are said to be *siblings*. The *size* $|t|$ of the tree $t$ is its number of nodes: $|t| = |\mathcal{V}_t|$. The *depth* of a node corresponds to the number of edges connecting it to the root plus one[1], and the depth of the tree is the maximum depth of its nodes. Finally, we introduce a couple of definitions that will be useful in the following.

[1]Note that the depth of the root node is one.

**Definition 5 (Balanced tree).** *A* perfectly depth-balanced tree *of order $h$ is a tree where the depth of each leaf node is $h$. Perfectly depth-balanced trees are also called* balanced trees *below.*

**Definition 6 (Branching cardinality).** *We define the* branching cardinality *of the tree $t$, noted branch($t$), as its number of leaf nodes minus one. More formally, for the tree $t = (\mathcal{V}_t, \mathcal{E}_t)$ with $\mathcal{V}_t = (v_1, \ldots, v_{|t|})$, branch($t$) is given by;*

$$branch(t) = \sum_{i=1}^{|t|} \mathbf{1}(d^+(v_i) = 0) - 1,$$

*where $\mathbf{1}(.)$ is a binary function equal to one if its argument is true, and zero otherwise.*

This terminology stems from the observation that this quantity also corresponds to the sum, over the non-leaf nodes of the tree, of their numbers of children minus one. It therefore measures how many extra branchings there are compared to a linear tree, which has branching cardinality 0. These definitions are illustrated in Figure 3.3.

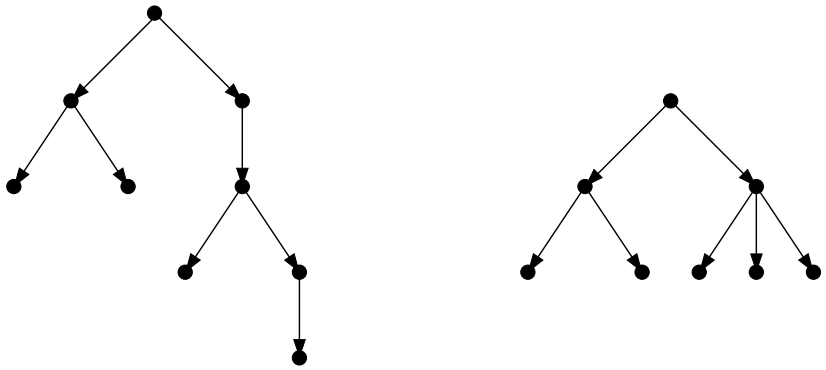

Figure 3.3: Left: a tree $t_1$ of depth 5 with $|t_1| = 9$ and branch$(t_1) = 3$. Right: a balanced tree $t_2$ of order 3 with $|t_2| = 8$ and branch$(t_2) = 4$. Top nodes are root nodes, bottom nodes are leaf nodes.

The remaining of the paper introduces kernel functions between labeled directed graphs based on the detection in the graphs of patterns corresponding to labeled trees. To lighten notations, we simply refer below to labeled directed graphs and labeled trees as graphs and trees.

## 3.2 Kernel definition

This section introduces a general class of graph kernel based on the detection, in the graphs, of patterns corresponding to particular tree structures. We start by defining precisely this notion of tree-pattern.

**Definition 7 (Tree-pattern).** *Let a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ and a tree $t = (\mathcal{V}_t, \mathcal{E}_t)$, with $\mathcal{V}_t = (n_1, \ldots, n_{|t|})$. A $|t|$-uple of vertices $(v_1, \ldots, v_{|t|}) \in \mathcal{V}_G^{|t|}$ is a* tree-pattern *of $G$ with respect to $t$, which we denote by $(v_1, \ldots, v_{|t|}) = pattern(t)$, if and only if the following holds:*

$$\begin{cases} \forall i \in [1, |t|], & l(v_i) = l(n_i)\,, \\ \forall (n_i, n_j) \in \mathcal{E}_t, & (v_i, v_j) \in \mathcal{E}_G \wedge l\big((v_i, v_j)\big) = l\big((n_i, n_j)\big)\,, \\ \forall (n_i, n_j), (n_i, n_k) \in \mathcal{E}_t, & j \neq k \iff v_j \neq v_k\,. \end{cases}$$

In other words a tree-pattern is a combination of graph vertices that can be arranged in a particular tree structure, according to the labels and the connectivity properties of the graph. Note from this definition that vertices of the graph are allowed to appear several times in a tree-pattern, under the condition that siblings nodes of the corresponding tree are associated to distinct vertices of the graphs. We now introduce a functional to count occurrences of these patterns.

**Definition 8 (Tree-pattern counting function).** *A* tree-pattern counting function *returning the number of times a tree-pattern occurs in a graph is defined for the tree $t$ and the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, $\mathcal{V}_G = (v_1, \ldots, v_{|\mathcal{V}_G|})$, as*

$$\psi_t(G) = \Big|\big\{(\alpha_1, \ldots, \alpha_{|t|}) \in [1, |\mathcal{V}_G|]^{|t|} : (v_{\alpha_1}, \ldots, v_{\alpha_{|t|}}) = pattern(t)\big\}\Big|.$$

*A restriction of $\psi_t$ to patterns rooted in a specified vertex $v$ is given by*

$$\psi_t^{(v)}(G) = \Big|\big\{(\alpha_1, \ldots, \alpha_{|t|}) \in [1, |\mathcal{V}_G|]^{|t|} :\ (v_{\alpha_1}, \ldots, v_{\alpha_{|t|}}) = pattern(t) \wedge v_{\alpha_1} = v\big\}\Big|.$$

With this new definition at hand we can define a general graph kernel based on the detection of common tree-patterns in the graphs.

**Definition 9 (Tree-pattern graph kernel).** The tree-pattern graph kernel $K$ *is given for the graphs $G_1$ and $G_2$ by*

$$K(G_1, G_2) = \sum_{t \in \mathcal{T}} w(t) \psi_t(G_1) \psi_t(G_2),$$

*where $\mathcal{T}$ is a set of trees, $w : \mathcal{T} \to \mathbb{R}$ is a tree weighting functional and $\psi_t$ is the tree-pattern counting function of Definition 8.*

The kernel of Definition 9 is obviously positive definite since it can be written as a standard dot-product $K(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle$, where $\phi(G)$ is the mapping that maps any graph $G$ to the feature space indexed by the trees of the set $\mathcal{T}$ as $\phi(G) = \big(\sqrt{w(t)}\psi_t(G)\big)_{t \in \mathcal{T}}$. Figure 3.4 illustrates this mapping.

## 3.3 Examples of tree-pattern graph kernels

In a recent work, Ramon and Gärtner (2003) proposed a particular tree-pattern graph kernel fitting the general Definition 9. In this section, we propose two different kernels with explicit feature spaces and inner products, discuss their practical computation, and highlight their differences with the kernel of Ramon and Gärtner (2003).

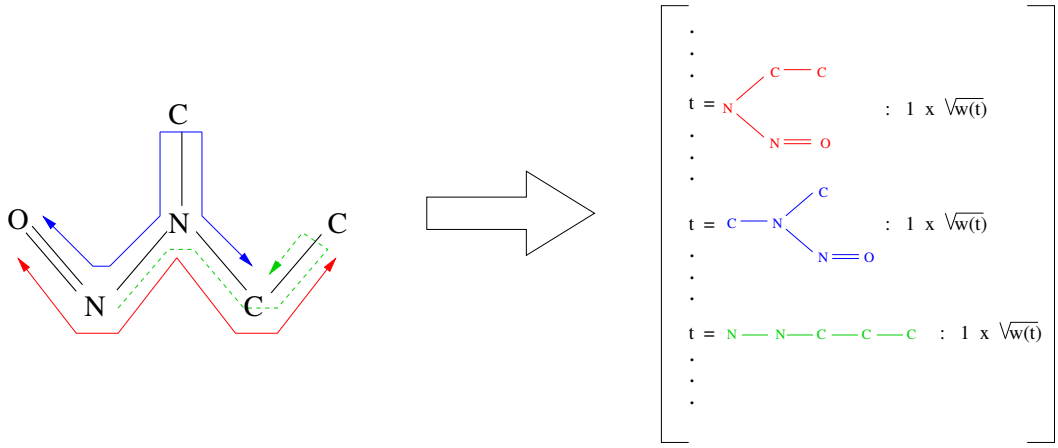


Figure 3.4: A molecular compound $G$ (left) and its feature space representation $\phi(G)$ (right). Note that the red and green trees are balanced. Note moreover that the green tree consists of a set of linearly connected atoms, which is known as *molecular fragment* in chemoinformatics. Note finally that the same $C$ atom appears in the 3rd and 5th positions in the tree-pattern corresponding to the green tree.

### 3.3.1 Kernels definition

According to Definition 9, two key elements enter in the definition of a tree-pattern graph kernel. Firstly, the set of trees $\mathcal{T}$ indexing the feature space the graphs are mapped to must be chosen. The kernels we consider in this section are based on the same feature space: the space indexed by the set of balanced trees of order $h$ introduced in Definition 5, labeled according to the graphs labeling alphabet $\mathcal{A}$. We will refer to this set as $\mathcal{B}_h$ in the following. Second, the tree weighting function $w$ must be defined. A natural way to define such a functional is to take into account the structure of the trees, and accordingly, we propose to relate the weight of a tree to its size or its branching cardinality. In particular we propose to consider the following kernels:

**Definition 10 (Size-based balanced tree-pattern kernel).** *For the pair of graphs $G_1$ and $G_2$, the* size-based balanced tree-pattern kernel of order $h$ *is defined as*

$$K^h_{Size}(G_1, G_2) = \sum_{t \in \mathcal{B}_h} \lambda^{|t|-h} \psi_t(G_1)\psi_t(G_2). \tag{3.1}$$

**Definition 11 (Branching-based balanced tree-pattern kernel).** *For the pair of graphs $G_1$ and $G_2$, the* branching-based balanced tree-pattern kernel of order $h$ *is defined as*

$$K^h_{Branch}(G_1, G_2) = \sum_{t \in \mathcal{B}_h} \lambda^{branch(t)} \psi_t(G_1)\psi_t(G_2). \tag{3.2}$$

Note that the depth of a tree is a lower bound on its size, attained for a tree consisting of a linear chain of vertices. For such a tree, at depth $h$, we have $|t| - h = \text{branch}(t) = 0$, and we see that the corresponding tree-patterns are given a unit weight in the kernels of Definitions 10 and 11. The complexity of a tree naturally increases with its size and branching cardinality, and the $\lambda$ parameter entering the kernel Definitions 10 and 11 has the effect of favoring tree-patterns depending on their degree of complexity. A value of $\lambda$ greater than one favors the influence of tree-patterns of increasing complexity over the trivial linear tree-patterns, while they are penalized by a value of $\lambda$ smaller than one. We can note, however, that while the size of a tree increases with its branching cardinality, the converse is not true. For any tree $t$ of depth $h$, we therefore always have $|t| - h \geq \text{branch}(t)$, and the tree weighting is more important in the size-based than in the branching-based kernel. In the case of balanced trees, this difference is particularly marked when the nodes with large out-degree are close to the root node. This is due to the fact that every leaf must be at depth $h$, and while the size of the tree necessarily increases by at least $h - 1$ along each path starting from the root, the branching cardinality does not[2]. The main difference in the feature space representations of the graphs is therefore induced by this particular type of tree-patterns, that can be interpreted as collections of regular subtree patterns merged in the root node. This suggests for instance that, for $\lambda < 1$, the branching-based formulation of the kernel may to some extent tolerate large, yet regular patterns, that would be strongly penalized in the size-based formulation. Figure 3.5 illustrates these tree weightings based on the size and branching cardinality.

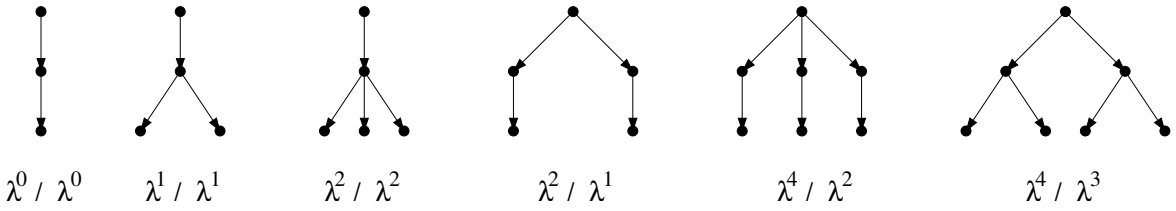


Figure 3.5: A set of balanced trees of order 3, together with their size-based (left) and branching-based (right) $\lambda$ weighting.

When $\lambda$ tends to zero, the complexity of the patterns is so penalized that only tree-patterns consisting of linear chains of graph vertices have non-vanishing weights, and the kernels of Definitions 10 and 11 boil down to a kernel based on the detection of common walks (Gärtner et al., 2003). More formally, if we define the set of walks of length $n$ of the graph $G$ as

$$\mathcal{W}_n(G) = \{(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1} : (v_i, v_{i+1}) \in \mathcal{E}_G,\ 0 \leq i \leq n-1\},$$

[2]At the extreme, we have $|t| = 1 + (h-1) \times d^+(r(t))$ vs $\text{branch}(t) = d^+(r(t)) - 1$.

and define for the graphs $G_1$ and $G_2$ the following walk-count kernel:

$$K^n_{\text{Walk}}(G_1, G_2) = \sum_{\substack{w_1 \in \\ \mathcal{W}_n(G_1)}} \sum_{\substack{w_2 \in \\ \mathcal{W}_n(G_2)}} \mathbf{1}(l(w_1) = l(w_2)), \tag{3.3}$$

where $\mathbf{1}(l(w_1) = l(w_2))$ is one if all pairs of corresponding edges and vertices are identically labeled in the walks $w_1$ and $w_2$, and zero otherwise, one easily gets that:

$$\lim_{\lambda \to 0} K^h_{\text{Size}}(G_1, G_2) = \lim_{\lambda \to 0} K^h_{\text{Branch}}(G_1, G_2) = K^{h-1}_{\text{Walk}}(G_1, G_2).$$

Increasing the value of $\lambda$ relaxes the penalization on complex subtree features, and can therefore be interpreted as introducing tree-patterns of increasing complexity in the walk-based kernel of Equation 3.3.

It should be noted finally that the parameters $h$ and $\lambda$ are directly related to the nature of the features representing the graphs and to their relative importance. Optimal values of the parameters are therefore likely to be dependent on the problem and data considered, and can hardly be chosen a priori. As an example, because of the variety of chemical compounds, the graphs considered in a chemical application can have a great structural diversity. This suggests that these parameters should be estimated from the data using, for example, cross-validation techniques.

### 3.3.2 Kernels computation

We now propose two factorization schemes to compute the kernels of Definitions 10 and 11. These factorizations are inspired by the dynamic programming (DP) algorithm proposed by Ramon and Gärtner (2003) to compute a slightly different graph kernel, discussed in the next subsection. The factorization relies on the following definition:

**Definition 12 (Neighborhood matching set).** *The* neighborhood matching set $\mathcal{M}(u, v)$ *of two graph vertices* $u$ *and* $v$ *is defined as*

$$\mathcal{M}(u, v) = \big\{ R \subseteq \delta^+(u) \times \delta^+(v) \mid \big(\forall (a, b), (c, d) \in R : a \neq c \wedge b \neq d\big) \\ \wedge \big(\forall (a, b) \in R : l(a) = l(b) \wedge l((u, a)) = l((v, b))\big)\big\}.$$

Each $R \in \mathcal{M}(u, v)$ consists of one or several pair(s) of neighbors of $u$ and $v$ that are identically labeled and connected to $u$ and $v$ by edges of the same label. It follows from Definition 5 that such an element $R$ corresponds to a pair of balanced tree-patterns of order 2 rooted in $u$ and $v$, found in the graph(s) $u$ and $v$ belong to. Moreover, provided $u$ and $v$ have the same label, these patterns correspond to the same balanced tree. We can state the following propositions, whose proofs are post-poned in Appendix 3.7:

**Proposition 13 (Size-based kernel computation).** *The order $h$ size-based tree-pattern kernel $K^h_{Size}$ of Definition 10 between two graphs $G_1$ and $G_2$ can be computed as:*

$$K^h_{Size}(G_1, G_2) = \frac{1}{\lambda^h} \sum_{u \in \mathcal{V}_{G_1}} \sum_{v \in \mathcal{V}_{G_2}} k_h(u, v), \tag{3.4}$$

*where $k_n, n = 1, \ldots, h$ is defined recursively by*

$$\begin{cases} k_1(u, v) = \lambda \mathbf{1}(l(u) = l(v)), \\ k_n(u, v) = \lambda \mathbf{1}(l(u) = l(v)) \displaystyle\sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} k_{n-1}(u', v'), \quad n = 2, \ldots, h. \end{cases}$$

**Proposition 14 (Branching-based kernel computation).** *The order $h$ branching-based tree-pattern kernel $K^h_{Branch}$ of Definition 11 between two graphs $G_1$ and $G_2$ can be computed as:*

$$K^h_{Branch}(G_1, G_2) = \sum_{u \in \mathcal{V}_{G_1}} \sum_{v \in \mathcal{V}_{G_2}} k_h(u, v), \tag{3.5}$$

*where $k_n, n = 1, \ldots, h$ is defined recursively by*

$$\begin{cases} k_1(u, v) = \mathbf{1}(l(u) = l(v)), \\ k_n(u, v) = \mathbf{1}(l(u) = l(v)) \displaystyle\sum_{R \in \mathcal{M}(u,v)} \frac{1}{\lambda} \prod_{(u',v') \in R} \lambda k_{n-1}(u', v'), \quad n = 2, \ldots, h. \end{cases}$$

Not surprisingly, Propositions 13 and 14 show that the kernels $K^h_{\text{Size}}$ and $K^h_{\text{Branch}}$ of Definitions 10 and 11 have the same complexity. More precisely, for the pair of graphs $G_1$ and $G_2$, it follows from (3.4) and (3.5) that this complexity is equal to the product of the sizes of $G_1$ and $G_2$, times the complexity of evaluating the functional $k_h$. In both cases, for the pair of graph vertices $u$ and $v$, evaluating $k_h(u, v)$ amounts to summing, over all possible matching of neighbors $R \in \mathcal{M}(u, v)$, a quantity expressed as a product of $|R|$ functionals $k_{h-1}$. The size of $\mathcal{M}(u, v)$, $|\mathcal{M}(u, v)|$, is maximal if all the neighbors of $u$ and $v$, as well as the edges that connect them to $u$ and $v$, are identically labeled. In that case we have

$$|\mathcal{M}(u, v)| = \sum_{k=1}^{\min(d^+(u), d^+(v))} A^k_{d^+(u)} A^k_{d^+(v)},$$

where $k$ ranges over the cardinality $|R|$ of the set of matching neighbors. If we let $d$ be an upper bond on the out-degree of the vertices of the graphs considered, it follows that $|\mathcal{M}(u, v)| \leq \sum_{k=1}^{d} (A^k_d)^2$ and we can derive the following worst case complexity

$$\mathcal{O}(K^h_{\text{Size}}(G_1, G_2)) = \mathcal{O}(K^h_{\text{Branch}}(G_1, G_2)) = |\mathcal{V}_{G_1}| \times |\mathcal{V}_{G_2}| \times (\sum_{k=1}^{d} k (A^k_d)^2)^{h-1}.$$

In the case of chemical compounds, we have $d = 4$. The factor $\sum_{k=1}^{d} k(A_d^k)^2$ equals 4336, and the complexity looks prohibitive. However this is only a worst-case complexity which is strongly reduced in practice because (i) the out-degree of the vertices is often smaller than 4[3], and (ii) the size of $\mathcal{M}(u, v)$ is reduced by the fact that vertices and edges can have distinct labels.

### 3.3.3 Relation to previous work

At this point, it is worth reminding the kernel formulation introduced by Ramon and Gärtner (2003) in order to highlight the differences with the kernels proposed in Definitions 10 and 11. In the context of graphs with labeled vertices and edges[4], at order $h$, the kernel introduced in Ramon and Gärtner (2003), that we denote by $K^h_{\text{Ramon}}$, is formulated as follows:

$$K^h_{\text{Ramon}}(G_1, G_2) = \sum_{u \in \mathcal{V}_{G_1}} \sum_{v \in \mathcal{V}_{G_2}} k_h(u, v),$$

where $k_n$ is defined by

$$\begin{cases} k_1(u, v) = \mathbf{1}(l(u) = l(v)) \\ k_n(u, v) = \mathbf{1}(l(u) = l(v))\ \lambda_u \lambda_v \displaystyle\sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} k_{n-1}(u', v'), \quad n = 2, \ldots, h. \end{cases}$$

It is clear that this kernel and the kernels of Definitions 10 and 11 have the same feature space. The main difference lies in the fact that in this formulation, a parameter $\lambda_v$ is introduced for each vertex $v$ of each graph. It can be checked that under this parametrization, each tree-pattern is weighted by the product of the parameters $\lambda_v$ associated to its internal nodes. In the special case where these parameters are taken equal to a single parameter $\lambda$, each pattern is therefore weighted by $\lambda$ raised to the power of its number of internal nodes. While this bears some similarity with the size-based weighting proposed in the kernel of Definition 10, we note for instance that the three leftmost trees of Figure 3.5 are identically weighted, namely by a factor $\lambda^2$. Moreover, the convergence to the walk-based kernel of Equation 3.3 observed when $\lambda$ tends to zero for the kernels of Definition 10 and 11 does not hold with this formulation.

## 3.4 Extensions

The kernels introduced in the previous section arise directly from the adaptation of the algorithm proposed in Ramon and Gärtner (2003). In this section we introduce

[3]For example, in the two datasets considered in our experiments in section 3.6, the average out-degree of the vertices is nearly 2 (2.14 for the first dataset, and 2.06 for the second one).

[4]The original formulation considered graphs with labeled vertices only, and the definition of the neighborhood matching set is refined in this paper in order to handle labeled edges.

two extensions to this initial formulation. First, we extend the branching-based kernel of Definition 11 to a feature space indexed by a larger, and more general, set of trees. Second, we propose to eliminate a set of noisy tree-patterns from the feature space.

### 3.4.1 Considering all trees

The DP algorithms of Section 3.3.2 recursively extend the tree-patterns under construction until they reach a specified depth. Because they are based on the notion of neighborhood matching sets introduced in Definition 12, these algorithms add at least one child to every leaf node of the patterns under extension at each step of the recursive process. When they reach the specified depth, the patterns are therefore balanced, and the choice of the feature space associated to the kernels of Definitions 10 and 11 was actually dictated by their computation.

Rather than focusing on features of a particular size, standard representations of molecules involve structural features of different sizes. A prominent example is that of molecular fingerprints (Ralaivola et al., 2005) that typically represent a molecule by its exhaustive list of fragments of length up to 8, where a fragment is defined as a linear succession of connected atoms (see Figure 3.4). In this section, we note that a slight modification of the DP algorithm of Proposition 14 generalizes the kernel of Definition 11 to a feature space indexed by the set of general trees up to a given depth, instead of the set of balanced-trees of the corresponding order. More precisely, if we let $\mathcal{T}_h$ be the set of trees of depth up to $h$, and if we define the *until-N extension* of the branching-based kernel of Definition 11 as

$$K_{\text{Branch}}^{\text{until-}h}(G_1, G_2) = \sum_{t \in \mathcal{T}_h} \lambda^{\text{branch}(t)} \psi_t(G_1) \psi_t(G_2), \tag{3.6}$$

we can state the following proposition, whose proof is postponed in Appendix 3.7.

**Proposition 15 (Until-N kernel computation).** *The* until-N extension $K_{Branch}^{until\text{-}h}$ *of the branching-based kernel of order* $h$ *of Definition 11 is given for the graphs* $G_1$ *and* $G_2$ *by*

$$K_{Branch}^{until\text{-}h}(G_1, G_2) = \sum_{u \in \mathcal{V}_{G_1}} \sum_{v \in \mathcal{V}_{G_2}} k_h(u, v),$$

*where* $k_n, n = 1, \ldots, h$ *is defined recursively by*

$$\begin{cases} k_1(u, v) = \mathbf{1}(l(u) = l(v)), \\ k_n(u, v) = \mathbf{1}(l(u) = l(v)) \left( 1 + \displaystyle\sum_{R \in \mathcal{M}(u,v)} \frac{1}{\lambda} \prod_{(u', v') \in R} \lambda k_{n-1}(u', v') \right), \quad n = 2, \ldots, h. \end{cases}$$

The computation given in Proposition 15 follows that of Proposition 14, and this until-N extension comes at no extra cost. The feature space corresponding to this extended kernel has nevertheless a much larger dimensionality than that of the original

branching-based kernel. Actually, because the set of trees $\mathcal{T}_h$ includes the set of balanced trees $\mathcal{B}_h$ as a special case, the feature space associated to the branching-based kernel is a sub-space of the feature space associated to its until-N extension. Figure 3.6 illustrate the different mappings. The behavior of this kernel with respect to $\lambda$ follows

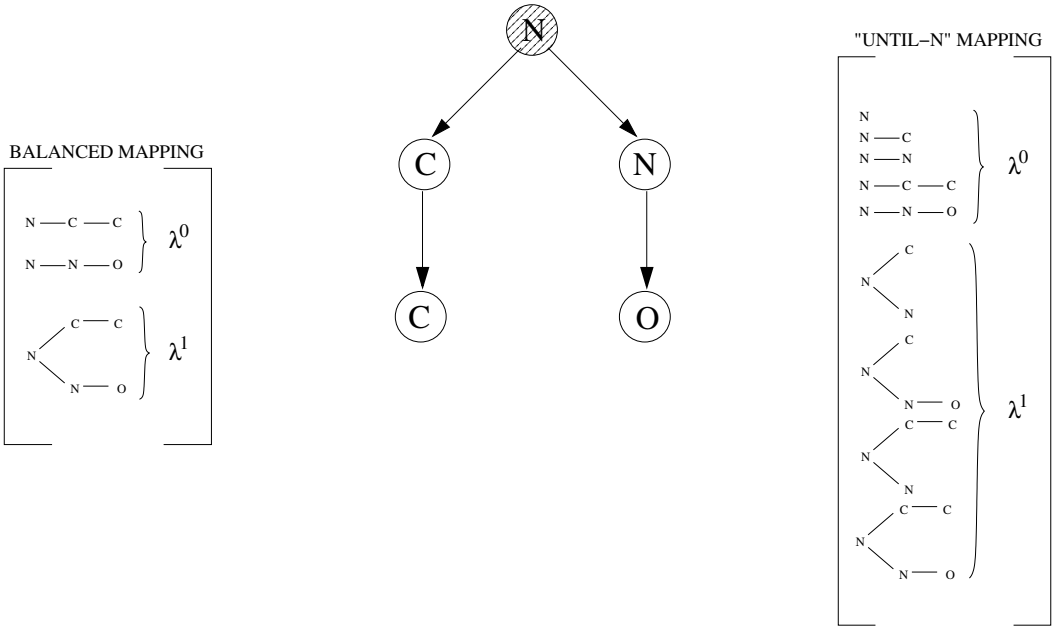


Figure 3.6: A graph $G$, and the set of balanced trees of order 3 (left) and general trees of depth up to 3 (right) for which a tree-pattern rooted in the dashed vertex is found in $G$, together with their kernel weighting $\lambda^{\text{branch}(t)}$.

that of the original branching-based kernel. In particular, when $\lambda$ tends to zero, the set of tree-patterns with non-vanishing weights reduces to linear chain of vertices and the kernel boils down to a kernel based on the detection of common walks of length up to $h-1$. More formally, one can easily check that, in this case:

$$\lim_{\lambda \to 0} K^{\text{until-}h}_{\text{Branch}}(G_1, G_2) = \sum_{n=0}^{h-1} K^{n}_{\text{Walk}}(G_1, G_2),$$

where $K^{n}_{\text{Walk}}$ is the kernel based on the detection of common walks of length $n$, defined in Section 3.3.1, Equation 3.3.

Finally, we note that this extension is not directly applicable to the size-based kernel of Definition 10 because of a slight difference in the computations of Propositions 13 and 14. Indeed, note from Proposition 13 that in order to get the $\lambda^{|t|-h}$ weighting of the tree $t$ proposed in Definition 10, the size-based kernel is initially computed from patterns weighted by their sizes, and is subsequently normalized by a factor $\lambda^{-h}$. As a result, while the above extension would still have the effect of extending the feature

space to the space indexed by trees of $\mathcal{T}_h$, this $\lambda^{-h}$ normalization would affect every tree-pattern regardless of their size, and the pattern weighting proposed in Definition 10 would be lost.

### 3.4.2 Removing tottering tree-patterns

The DP algorithms of Sections 3.3.2 and 3.4.1 enumerate balanced tree-patterns of order $h$ through the recursive extension of balanced tree-patterns of order 2 defined by neighborhood matching sets of pairs of vertices. According to Definition 12, the whole sets of neighbors of a pair of vertices enter in the definition of their neighborhood matching sets. As a result, it can be the case in a tree-pattern that a vertex appears simultaneously as the parent and a child of a second vertex. This phenomenon is the tree counterpart of a phenomenon observed in the context of walk-based graph kernels, where a random walk under extension could return to a visited vertex just after leaving it. This behavior was called *tottering* in Mahé et al. (2005), and following this terminology, we refer to a tree-pattern in which a vertex appears simultaneously as the parent and a child of a second vertex as a *tottering tree-pattern*. Figure 3.7 illustrates the tottering phenomenon.

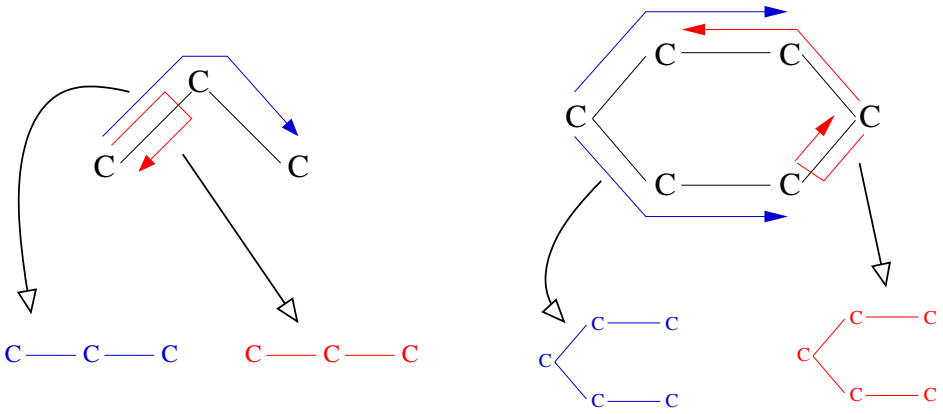


Figure 3.7: Left: tottering (red) and no-tottering (blue) walks. Right: tottering (red) and no-tottering(blue) tree-patterns.

In many cases these tree-patterns are likely to be uninformative features. In particular they are not proper subgraphs of the initial graphs. Even worse, the ratio of the number of tottering tree-patterns over the number of non-tottering tree-patterns quickly increases with the depth $h$ of the trees, suggesting that informative patterns corresponding to deep trees might be hidden by the profusion of tottering tree-patterns. In order to tackle this issue we now adapt an idea of Mahé et al. (2005) to filter out these spurious tottering tree-patterns in the kernels presented in Sections 3.2 and 3.3. Tottering can be prevented by adding constraints in the tree-pattern counting function, according to the following definition.

**Definition 16 (No-tottering tree-pattern counting function).** *From the tree-pattern counting function of Definition 8, a* no-tottering tree-pattern counting function *can be defined for the tree $t = (\mathcal{V}_t, \mathcal{E}_t)$, with $\mathcal{V}_t = (n_1, \ldots, n_{|t|})$, and the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, with $\mathcal{V}_G = (v_1, \ldots, v_{|\mathcal{V}_G|})$, as*

$$\psi_t^{NT}(G) = \Big|\Big\{(\alpha_1, \ldots, \alpha_{|t|}) \in [1, |\mathcal{V}_G|]^{|t|} : \quad (v_{\alpha_1}, \ldots, v_{\alpha_{|t|}}) = \mathit{pattern}(t) \\ \wedge (n_i, n_j), (n_j, n_k) \in \mathcal{E}_t \iff \alpha_i \neq \alpha_k\Big\}\Big|.$$

Following Definition 9, a graph kernel based on no-tottering tree-patterns can be defined from this no-tottering tree-pattern counting function.

**Definition 17 (No-tottering tree-pattern kernel).** *A graph kernel $K^{NT}$ based on no-tottering tree-patterns is given for the graphs $G_1$ and $G_2$ by*

$$K^{NT}(G_1, G_2) = \sum_{t \in \mathcal{T}} w(t) \psi_t^{NT}(G_1) \psi_t^{NT}(G_2), \tag{3.7}$$

*where $\mathcal{T}$ is a set of trees, $w : \mathcal{T} \to \mathbb{R}$ is a tree weighting functional and $\psi_t^{NT}$ is the no-tottering tree-pattern counting function of Definition 16.*

This latter definition therefore extends the tree-pattern kernel of Definition 9 to the no-tottering case. However, due to the additional constraints on the set of acceptable patterns, the DP framework based on neighborhood matching set described in Sections 3.3.2 and 3.4.1 does not hold any longer. In Mahé et al. (2005), the following graph transformation was introduced in order to filter tottering walks.

**Definition 18 (Graph transformation).** *For a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, we let its* transformed graph *$G' = (\mathcal{V}_{G'}, \mathcal{E}_{G'})$ be defined by:*

- $\mathcal{V}_{G'} = \mathcal{V}_G \cup \mathcal{E}_G$,
- $\mathcal{E}_{G'} = \{(v, (v,t)) \,|v \in \mathcal{V}_G, (v,t) \in \mathcal{E}_G\} \cup \{((u,v), (v,t)) \,|\, (u,v), (v,t) \in \mathcal{E}_G, u \neq t\}$,

*and labeled as follows:*

- *for a node $v' \in \mathcal{V}_{G'}$ the label is either $l(v') = l(v')$ if $v' \in \mathcal{V}_G$, or $l(v') = l(v)$ if $v' = (u,v) \in \mathcal{E}_G$,*
- *for an edge $e' = (v'_1, v'_2)$ between two vertices $v'_1 \in \mathcal{V}_G \cup \mathcal{E}_G$ and $v'_2 \in \mathcal{E}_G$, the label is simply given by $l(e') = l(v'_2)$.*

This graph transformation is illustrated in Figure 3.8 for the graph corresponding to the chemical compound of Figure 3.2. Based on this graph transformation, Mahé et al. (2005) proved that there is a bijection between the set of no-tottering walks of a graph and the set of walks of its transformed graph that start on a vertex corresponding to a vertex of the original graph. In a similar way, we show below that there is a bijection between the set of no-tottering tree-patterns found in a graph and the set of tree-patterns found in its transformed graph rooted in a vertex corresponding to a vertex of the original graph. This is summarized in the following proposition, which proof is postponed in Appendix 3.7.

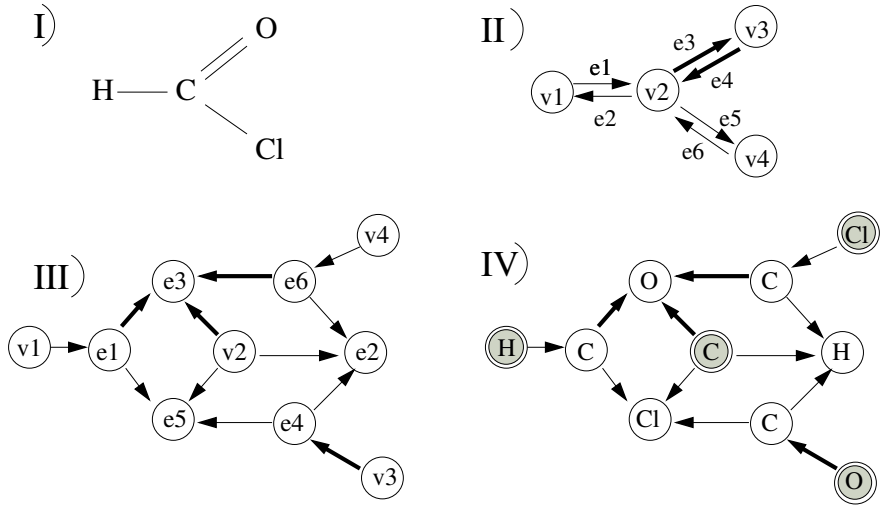


Figure 3.8: The graph transformation. I) The original molecule. II) The corresponding graph $G = (\mathcal{V}_G, \mathcal{E}_G)$. III) The transformed graph. IV) The labels on the transformed graph. Note that different widths stand for different edges labels, and gray nodes are the nodes belonging to $\mathcal{V}_G$.

**Proposition 19.** *If we let $G'_1$ (resp. $G'_2$) be the transformed graph of $G_1$ (resp. $G_2$), the no-tottering tree-pattern kernel of Definition 17 is given by*

$$\begin{aligned} K^{NT}(G_1, G_2) &= \sum_{t \in \mathcal{T}} w(t) \psi_t^{NT}(G_1) \psi_t^{NT}(G_2) \\ &= \sum_{t \in \mathcal{T}} w(t) \psi_t^{\{V_{G_1}\}}(G'_1) \psi_t^{\{V_{G_2}\}}(G'_2), \end{aligned}$$

*where, if $G'$ is the transformed graph of $G$ given by Definition 18, $V_G \subset \mathcal{V}_{G'}$ is the set of vertices of $G'$ corresponding to the vertices of $G$, and $\psi_t^{\{v_1,\ldots,v_n\}}(G) = \sum_{i=1}^{n} \psi_t^{(v_i)}(G)$.*

This proposition shows that we can compute no-tottering extensions of the kernels of Definitions 10 and 11, and of the until-N kernel extension of Equation 3.6, using the graph transformation of Definition 18 and the original DP algorithms of Sections 3.3.2 and 3.4.1. However, this operation comes at the expense of an increase in the cost of computing the kernel. More precisely, by definition of the graph transformation, we have $|\mathcal{V}_{G'}| = |\mathcal{V}_G| + |\mathcal{E}_G|$. Moreover, as noticed by Mahé et al. (2005), the maximum out-degree of the vertices of the transformed graph is equal to that of the original graph. As a result, the worst case complexity of evaluating the functional $k_h(u, v)$ of Propositions 13, 14 and 15 is the same if $u$ and $v$ belong to $\mathcal{V}_{G'_1}$ and $\mathcal{V}_{G'_2}$, or $\mathcal{V}_{G_1}$ and $\mathcal{V}_{G_2}$. It follows that for the graphs $G_1$ and $G_2$ we have

$$\mathcal{O}\big(K^{NT}(G_1, G_2)\big) = \frac{(|\mathcal{V}_{G_1}| + |\mathcal{E}_{G_1}|)(|\mathcal{V}_{G_2}| + |\mathcal{E}_{G_2}|)}{|\mathcal{V}_{G_1}||\mathcal{V}_{G_2}|} \quad \mathcal{O}\big(K(G_1, G_2)\big),$$

where $K$ is one of the kernels given in Equations 3.1, 3.2 and 3.6, and $K_{NT}$ is its no-tottering extension of Definition 17.

## 3.5 Implementation

In this section we detail the implementation of the original size-based and branching-based kernel computations given Propositions 13 and 14, and of the extensions of Section 3.4. The original size-based and branching-based kernels are computed by a DP algorithm that consists, for the pair of graphs $G_1$ and $G_2$ with $\mathcal{V}_{G_1} = \{v_1, \ldots, v_{|\mathcal{V}_{G_1}|}\}$ and $\mathcal{V}_{G_2} = \{u_1, \ldots, u_{|\mathcal{V}_{G_2}|}\}$, in computing a set of $h$, initially null, matrices $M_1, \ldots, M_h$ of size $|\mathcal{V}_{G_1}| \times |\mathcal{V}_{G_2}|$ defined by $M_n[i,j] = k_n(v_i, u_j)$. The corresponding kernel $K$ is then obtained by summing the entries of the matrix $M_h$, that is $K(G_1, G_2) = \sum_{i=1}^{|\mathcal{V}_{G_1}|} \sum_{j=1}^{|\mathcal{V}_{G_2}|} M_h[i,j]$.

From Propositions 13 and 14, the matrix $M_1$ is obtained directly by setting to $\lambda$, for the size-based kernel, or 1, for the branching-based kernel, the entries corresponding to pairs of vertices identically labeled. For $n = 2, \ldots, h$, the matrix $M_n$ can be obtained from the matrix $M_{n-1}$ using the following computation. For each pair of vertices $v_i \in \mathcal{V}_{G_1}$ and $u_j \in \mathcal{V}_{G_2}$ such that $l(v_i) = l(u_j)$:

- for $d = 1, \ldots, \min(d^+(v_i), d^+(u_j))$:
  - compute $N_1$, the set of possible arrangements of $d$ neighbors of $v_i$
  - compute $N_2$, the set of possible arrangements of $d$ neighbors of $u_j$
  - for each pair of arrangements $(n_1, \ldots, n_d) \in N_1$ and $(n'_1, \ldots, n'_d) \in N_2$:
    - if $l(n_k) = l(n'_k) \wedge l((v_i, n_k)) = l((u_j, n'_k)),\ k = 1, \ldots, d$:
      *# that is, if the pair of arrangements respects the constraints of* $\mathcal{M}(v_i, u_j)$
      - for the size-based kernel :
        $$M_n[i,j] \mathrel{+}= \lambda \prod_{k=1}^{d} M_{n-1}[\mathrm{ind}(n_k), \mathrm{ind}(n'_k)]$$
      - for the branching-based kernel :
        $$M_n[i,j] \mathrel{+}= \lambda^{d-1} \prod_{k=1}^{d} M_{n-1}[\mathrm{ind}(n_k), \mathrm{ind}(n'_k)]$$
        *# where, for the vertex $n$ of the graph $G$, ind$(n)$ returns the index of $n$ in $\mathcal{V}_G$*

In the case of the size-based kernel, the entries of $M_h$ must eventually be normalized by a factor $\lambda^{-h}$ before they are summed to return the kernel.

This pseudo-code implements the computations of Propositions 13 and 14. However, we added to this implementation an extra constraint in the definition of the neighborhood matching set. Instead of simply requiring that matching arrangements of neighbors consist of pairs of identically labeled vertices and edges, we also require

that their vertices follow a topological order. This has the effect of preventing repetitions of dimensions of the feature space indexed by trees identical up to a permutation of their sets of sibling nodes. In the case where vertices correspond to atoms, a natural ordering is defined by their atomic number, and, as a result, we replaced the condition

$$\text{if } l(n_k) = l(n'_k) \wedge l((v_i, n_k)) = l((u_j, n'_k)),\ k = 1, \ldots, d$$

in the above pseudo-code by

$$\begin{aligned}\text{if } l(n_k) = l(n'_k) \wedge l((v_i, n_k)) = l((u_j, n'_k)),\ k = 1, \ldots, d \\ \wedge\, AN(n_k) < AN(n_{k+1}),\ k = 1, \ldots, d-1,\end{aligned}$$

where $AN(v)$ returns the (integer) atomic number corresponding to the atom type encoded in the label of the vertex $v$. As a result, sibling nodes in the trees indexing the feature space are ordered according to their atomic number, which limits the dimensionality of the feature space. This is illustrated in Figure 3.9.

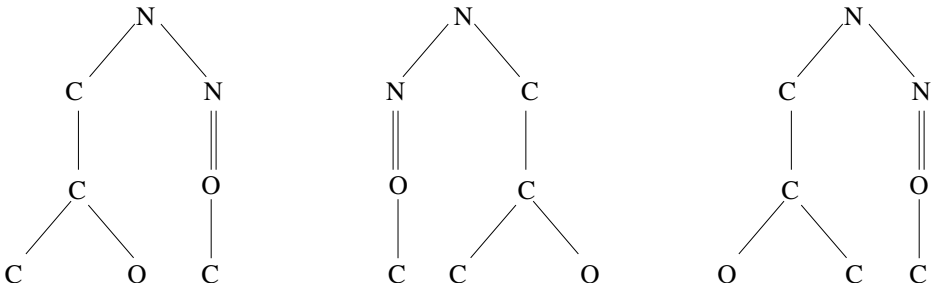


Figure 3.9: A set of balanced trees identical up to permutations of their sets of sibling nodes. Note that $AN(C) = 14$, $AN(N) = 15$ and $AN(O) = 16$. The tree in the center (resp. right) does not respect the topological ordering because of the set of sibling nodes $(N, C)$ at depth 2 (resp. $(O, C)$ at depth 4). The leftmost tree respects the atomic number ordering and is therefore the only one, among these three trees, to index the feature space.

The extensions to this initial formulation introduced in Sections 3.4.1 and 3.4.2 are easily included in this computation. In the above pseudo-code, the until-N extension simply consists in adding 1 to $M_n[i, j]$ after the loop on the size $d$ of potential matching sets of neighbors. As for the no-tottering extension, it formulates as a preprocessing step of graph transformation, after which the above pseudo-code can be applied based on the transformed graphs. However, in that case, the summation of the matrix $M_h$ returning the kernel must be restricted to the entries corresponding to the pairs of vertices of the transformed graphs that are associated to the vertices of the original ones.

This implementation is clearly not optimal, but in practice, because the average out-degree of the graphs we consider in the applications of the next section is roughly two, computations times were acceptable. Many room is nevertheless left for improvement, in particular in the way of matching the neighbors of a pair of vertices. Indeed, instead of exhaustively list all possible combinations of neigbhors of each vertex, and then evaluate their pairwise matching, it should be possible to directly define their neighborhood matching set, according to Definition 12. Note finally that this algorithm was implemented in C++ and is available within the open-source ChemCpp toolbox[5], dedicated to the computation of kernel functions between molecular compounds.

## 3.6 Experiments

We now turn to the experimental section. The problem we consider is a binary classification task consisting in discriminating toxic from non-toxic molecules. Our main goal is to assess the relevance of tree-patterns graph kernels over their walk-based counterparts for this type of chemical applications. To do so, recall from section 3.3.1 that in the proposed kernels, the influence of the tree-patterns is controlled by the parameter $\lambda$. When $\lambda$ tends to zero, the kernels converge to kernels based on the count of common walks in the graphs (Gärtner et al., 2003). For increasing $\lambda$, tree-patterns of increasing complexity are taken into account with increasing weight in the kernels. One can therefore study the relevance of tree-patterns by studying how the performance of the kernels evolves with $\lambda > 0$, and checking whether it improves over their walk-based counterpart obtained for $\lambda = 0$.

The first step towards this goal is to evaluate the kernels of Definitions 10 and 11, and therefore the original formulation presented in Ramon and Gärtner (2003). In a second step, we want to validate the extensions to these kernels proposed in sections 3.4.1 and 3.4.2. On the one hand we will compare the results obtained with the until-N extension of the branching-based kernel (3.6) to its initial formulation (3.2), and on the other hand we will compare the results obtained with the no-tottering extensions (3.7) of the size-based, branching-based, and until-N branching-based kernels to their original formulations. Because our interest here is to get insights about the behavior of the different kernels, we report experimental results for varying values of the parameters entering their definition, namely the order $h$ of the patterns, and the pattern weighting parameter $\lambda$. In real-world applications one should of course design a procedure to select the best parameters from the date.

The classification experiments described below were carried out with a support vector machine based on the different kernels tested. Each kernel was implemented in C++ within the open-source ChemCpp toolbox, and we used the open-source Python machine learning package PyML[6] to perform SVM classification. The SVM prediction

[5]Available at `http://chemcpp.sourceforge.net`

[6]Available at `http://pyml.sourceforge.net`

is obtained by taking the sign of a score function. However, by varying this zero decision threshold, it is possible to compute the evolution of the true positive rate versus the false positive rate in a curve known as the Receiver Operating Characteristic (ROC) curve. The area under this curve, known as AUC for Area Under the ROC Curve, is often considered to be a safer indicator of the quality of a classifier than its accuracy (Fawcett, 2003), being 1 for an ideal classifier, and 0.5 for a random classifier. The results presented below are averaged AUC values obtained for 10 repetitions of a 5-fold cross-validation process. Within each cross-validation fold, the "$C$" soft-margin parameter of the SVM was optimized over a grid ranging from $10^{-3}$ to $10^{3}$, using an internal cross-validation method implemented in PyML.

We considered two public datasets of chemical compounds in our experiments. Both gather results of mutagenicity assays, and while the first one (King et al., 1996) is a standard benchmark for evaluating chemical compounds classification, the second one (Helma et al., 2004) was introduced more recently. The first dataset contains 188 chemical compounds tested for mutagenicity on *Salmonella typhimurium*. The molecules of this dataset belong to the family of aromatic and hetero-aromatic nitro compounds, and they are split into two classes: 125 positive examples with high mutagenic activity (positive levels of log mutagenicity), and 63 negative examples with no or low mutagenic activity. The second database considered consists of 684 compounds classified as mutagens or non-mutagens according to a test known as the *Salmonella*/microsome assay. This dataset is well balanced with 341 mutagens compounds for 343 non-mutagens ones. Note that although the biological property to be predicted is the same, the two datasets are fundamentally different. While King et al. (1996) focused on a particular family of molecules, this dataset involves a set of very diverse chemical compounds, qualified as *noncongeneric* in the original paper. To predict mutagenicity, the model therefore needs to solve different tasks : in the first case it has to detect subtle differences between homogeneous structures, while in the second case it must seek regular patterns within a set of structurally different molecules.

### 3.6.1 First dataset

**Tree-patterns Vs walk-patterns:**
Figure 3.10 shows the results obtained for the size-based (left) and branching-based (right) kernels of Definitions 10 and 11. Each curve represents the evolution, for $0 \leq \lambda \leq 1$, of the AUC obtained from patterns of a given order $h$ taken between 2 and 10.

Because the corresponding AUC values start by increasing with $\lambda$, we can note from Figure 3.10 (left) that the introduction of tree-patterns is beneficial to the size-based kernel for patterns of order greater than two. In the case of the branching-based kernel, Figure 3.10 (right) suggests that this is only true for patterns of order greater than 2 and smaller than 6, but Figure 3.11 shows that, based on smaller values of $\lambda$, this is still the case for patterns up to order 7. Taken together, Figures 3.10 and 3.11 show that the optimal AUC values obtained with the size- and branching-based kernels for

patterns of order 2 to 7 are globally similar. Interestingly however, the corresponding $\lambda$ values are systematically smaller in the case of the branching-based kernel. This is due to the fact that, as noted in section 3.3.1, the size-based penalization is stronger than the branching-based penalization. As a result, optimal $\lambda$ values observed using the size-based kernel are shifted towards zero using the branching-based kernel.

We can also note from Figures 3.10 and 3.11 that optimal values of $\lambda$ tend to decrease for increasing $h$. This is probably due to the fact that the number of tree-patterns increases exponentially with $h$, and, as a result, the kernels need to limit their individual influence. Actually, we observe that higher order patterns, with $h > 7$, can only be considered for sufficiently small values of $\lambda$. For example, we note that the size-based kernel computation does not converge if we consider patterns of order 10 and $\lambda$ greater than 0.15. In the case of branching-based kernel, due to the weaker pattern penalization, this phenomenon is even emphasized, and in that case, $10^{-4}$ is the largest value acceptable for $\lambda$. This difference in the way to penalize the patterns probably explains the fact that while a slight improvement over the walk-based kernel can be observed in the case of the size-based kernel when $h$ is greater than 7 (Figure 3.10, left), the performance systematically decreases with the branching-based kernel (Figure 3.11).

Additionally, we note that because the size- and branching-based penalization of balanced trees of order 2 is the same, the results obtained for $h = 2$ are identical with the two kernels. Surprisingly however, no improvement over the walk-based baseline is observed, which suggests that in this case, the tree-patterns do not bring additional information to that contained in the walk features, that consist here of simple pairs of connected atoms.

In conclusion, these experiments demonstrate the improvement of the tree-patterns graph kernels over their walk-based counterparts. The impact of the tree-patterns is particularly marked for patterns of order 3 and 4, where the two kernels improve by more than 3% the AUC of the corresponding walk-based kernel. For patterns of increasing order, this figure gradually decreases, and for patterns of order greater than 7, it drops to 1 % in the case of the size-based kernel, while no more improvement is observed with the branching-based kernel. In both cases, optimal results are obtained for patterns of order 4, with AUC values of 95.3% and 95.0%. Finally, it is worth noting the combinatorial explosion in the number of patterns for large orders, which in practice limits the acceptable values of $\lambda$ to small values.

**Until-N extension:**
Figure 3.12 presents the results of the until-N extension (3.6) of the branching-based kernel (3.2). The figure on the left-hand side, showing the evolution of the AUC for $2 \leq h \leq 10$ and $0 \leq \lambda \leq 1$, corresponds to that on the right-hand side of Figure 3.10. The figure on the right-hand side plots these AUC values versus corresponding values obtained using the original kernel (3.2).

We can first notice strong similarities between the curve in the left-hand side and its original kernel counterpart. This is confirmed in the right-hand curve where all the

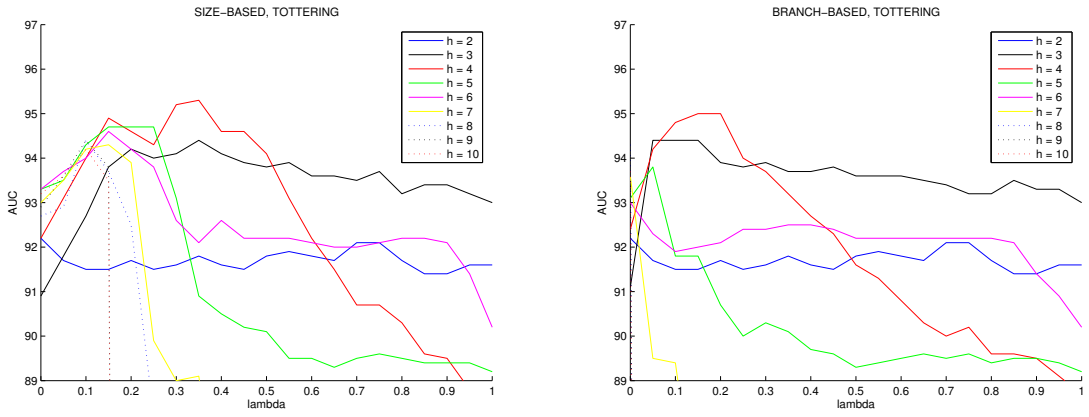


Figure 3.10: First dataset. Evolution of the AUC with respect to $\lambda$ at different orders $h$. Left: size-based kernel (3.1) ; Right: branching-based kernel (3.2).

points lie near the diagonal line that represents the equivalence between the two kernels. The fact that the differences between the two kernel formulations are barely noticeable is quite surprising since their associated feature spaces are intuitively quite different. In section 3.4.1, we mentioned that the feature space associated to the branching-based kernel is actually a subspace of the feature space associated to its until-N extension. As a result, Figure 3.12 suggests that the extra features related to the until-N extension do not bear additional information into the kernel. This hypothesis seems to be confirmed by the fact that the differences between corresponding walk-based kernels, observed for $\lambda = 0$, are not significant neither. This might be explained by the fact that the dimensions of the corresponding feature space are probably strongly correlated due to the relation of inclusion existing between trees and walks patterns of orders $n$, and those of order $n + 1$. Another possible explanation for the lack of improvement of the until-N extension lies of course in the difficulty of learning in high dimension, suggesting that discriminating patterns of a given order are lost within the flood of patterns of greater orders taken into account by this until-N extension.

**No-tottering extension:**
Figures 3.13, 3.14 and 3.15 respectively show the results of the no-tottering extension (3.7) of the size-based (3.1), branching-based (3.2), and until-N branching-based kernels (3.6). The curves on the left-hand side show the evolution of AUC for $2 \leq h \leq 10$ and $0 \leq \lambda \leq 1$, and the curves on the right-hand side plot these AUC values versus corresponding values obtained using the original kernels.

If we compare the results of the no-tottering extensions of the size-based and branching-based kernels (Figures 3.13 and 3.14), we can first note that the the introduction of tree-patterns is now systematically beneficial for $h > 2$ in both cases. Moreover, we note that the kernel computations remain feasible for $h = 10$ and $\lambda = 1$, which means that the no-tottering extension limits the combinatorial explosion we observed with the original formulation. While optimal results were obtained for $h = 4$

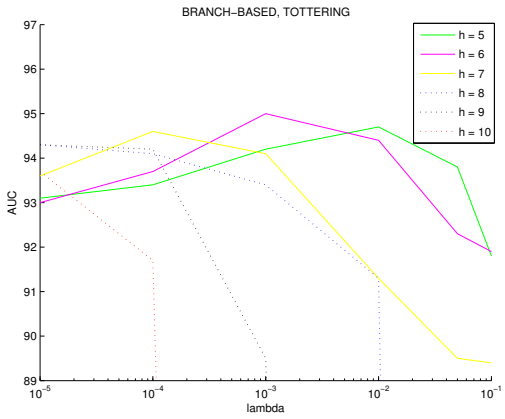


Figure 3.11: First dataset, branching-based kernel (3.2) . Evolution of the AUC at different orders $h$ for small values of $\lambda$.

using the original kernels, we observe that here, in both cases, the performance gradually increases from $h = 3$ to an optimum value obtained for $h = 8$. At a given order, we note that the optimal AUC values obtained with the two kernels are similar, and that the corresponding $\lambda$ value is smaller in the case of the branching-based kernel, which is consistent with the observations made in the previous section. Optimal AUC values are close to 96.5% and improve over the values around 95% observed with the initial formulation. Importantly, we note that these optimal values are obtained using parametrizations of the kernels that lead to a combinatorial explosion in their initial formulation. Finally, from the fact that almost all points lie above the diagonal in the right-hand curves, we can draw the conclusion that the no-tottering extension has almost consistently a positive influence on the classification in both cases. It is worth noting however that, even though the introduction of no-tottering tree-patterns was shown to be beneficial, part of the overall improvement over their tottering counterparts is due to the no-tottering extension itself, since no-tottering walk-based kernels, observed for $\lambda = 0$, already improve significantly over their tottering counterparts, especially for high order patterns.

We now turn to Figure 3.15 and the no-tottering extension (3.7) of the until-N branching-based kernel (3.6). We can first notice that conclusions similar to those related to the no-tottering extension of the branching-based kernel can be drawn: an improvement over the corresponding walk-based kernel is systematically observed for tree-patterns of order greater than 2, the kernel behaves more nicely (no combinatorial explosion), and the no-tottering extension consistently improves over the initial until-N branching-based kernel (right-hand curve). Interestingly however, we note that optimal results obtained for $4 \leq h \leq 10$ tend to converge to an optimal value around 95.5% (between 95.3 and 95.9%) for a $\lambda$ value around 0.05. While this global optimum is not as good as the overall optimal result obtained with the no-tottering branch-based

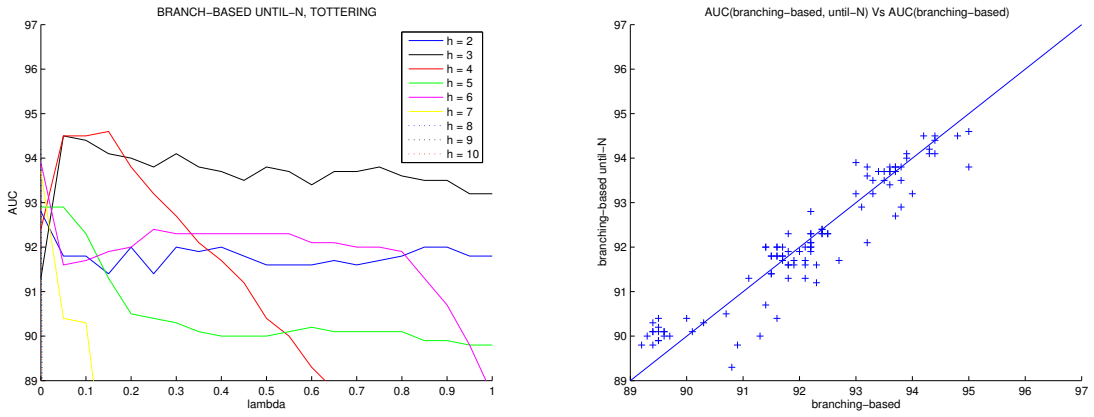


Figure 3.12: First dataset, until-N extension. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$, for the until-N extension (3.6) of the branching-based kernel (3.2). Right: AUC values Vs original AUC values.

kernel (Figure 3.14), it still remains competitive (95.5% Vs 96.5%). This observation contrasts with the the results obtained with the until-N extension in the tottering case, where patterns of a given order seemed to be lost in the amount of patterns of greater orders taken into account by the kernel. This is due to the fact the the no-tottering extension limits the number of patterns to be detected, and suggests that patterns of different orders can now be considered simultaneously in the kernel. This fact therefore suggests that in the no-tottering case, the until-N extension can help solving the problem of pattern order selection by taking a maximal pattern order large enough (here, $h > 4$).

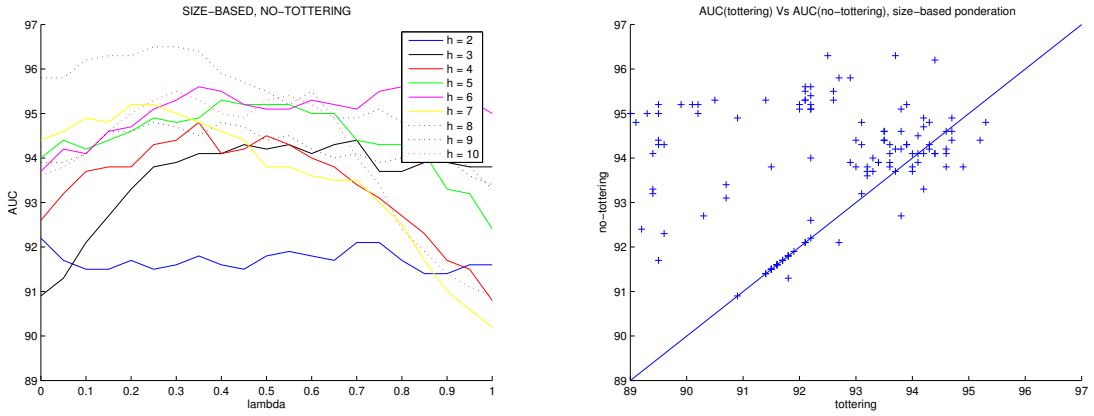


Figure 3.13: First dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the size-based kernel (3.1). Right: no-tottering AUC values Vs original AUC values.

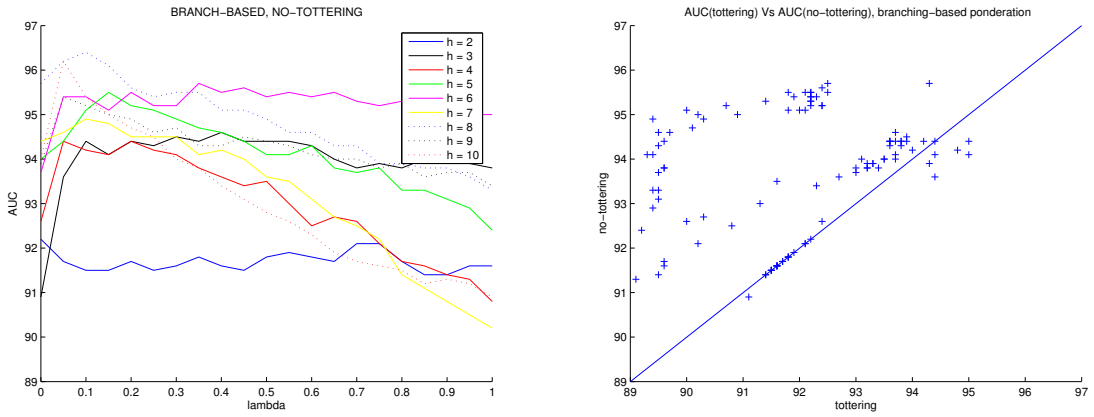


Figure 3.14: First dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the branching-based kernel (3.2). Right: no-tottering AUC values Vs original AUC values.

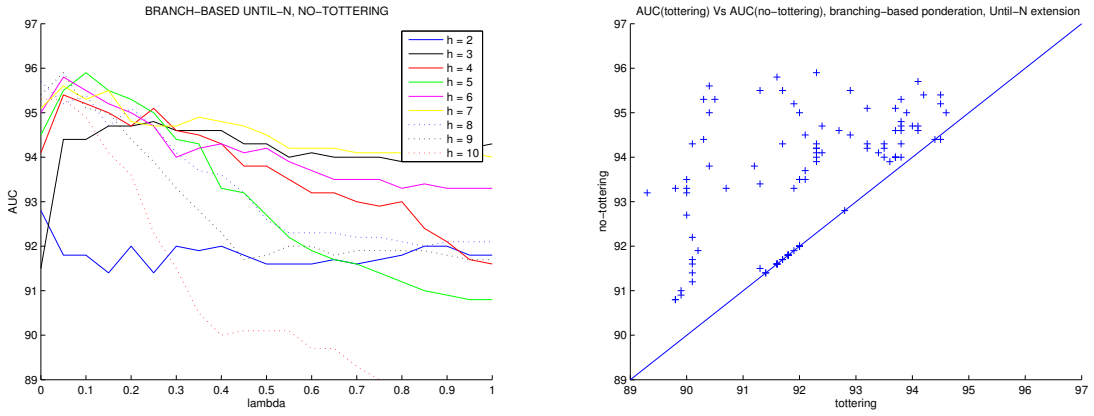


Figure 3.15: First dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the until-N branching-based kernel (3.6). Right: no-tottering AUC values Vs original AUC values.

### 3.6.2 Second dataset

In this section, we apply the same analysis to the second dataset.

**Tree-patterns Vs walk-patterns:**
Figure 3.16 shows the results obtained with the original size-based (3.1) and branching-based (3.2) kernels. Several observations are consistent with those we drew with the fist dataset. First, the introduction of tree-patterns has in both cases a positive influence on the classification, and is particularly marked for patterns of limited order (up to a relative improvement of 12% for $h = 2$, and 4.5% for $h = 3$). Moreover, optimal values of the $\lambda$ parameter are smaller in the case of the branching-based kernel, they decrease for increasing $h$, and quickly lead to a combinatorial explosion for high-order patterns. Finally, we note that, in both cases, optimal AUC values are around 84%, and are obtained for patterns of order 3 and 4, which is similar to the optimal order observed for the first dataset. However, we can note the interesting difference that here, tree-patterns of order 2 improve dramatically the results over their walk counterparts, which suggests that different molecular features are to be detected in both datasets.

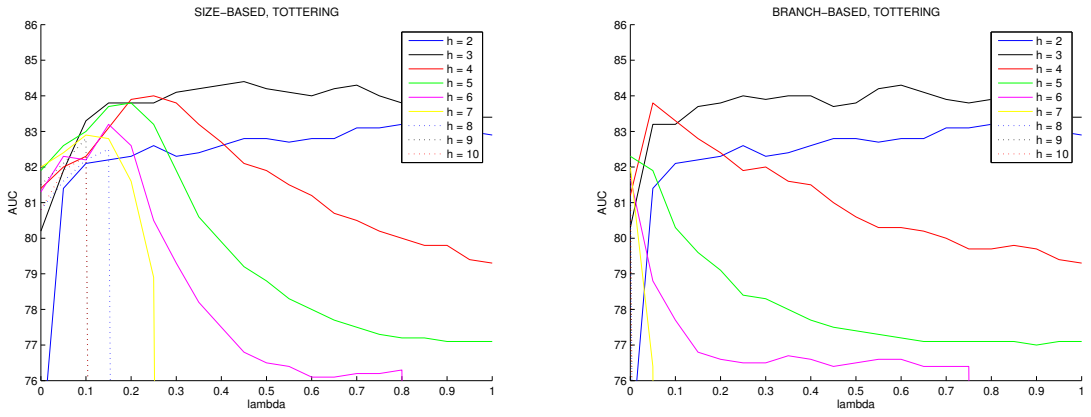


Figure 3.16: Second dataset. Evolution of the AUC with respect to $\lambda$ at different orders $h$. Left: size-based kernel (3.1) ; Right: branching-based kernel (3.2).

**Until-N extension:**
Figure 3.17 shows the results obtained with the until-N extension (3.6) of the branching-based kernel (3.2). Here again, observations are consistent with the first dataset. In particular, we can note that the results obtained with and without the until-N extension are very similar, and this fact is even more pronounced here. This second evidence confirms that the until-N extension is of little use in the original formulation of the kernel, most probably because patterns of a given order are drowned within the amount of patterns of greater orders.

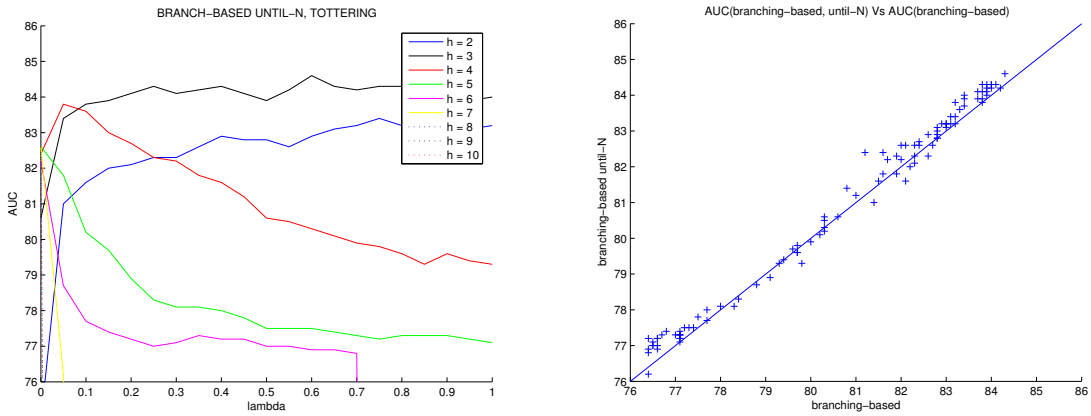


Figure 3.17: Second dataset, until-N extension. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the until-N extension (3.6) of the branching-based kernel (3.2). Right: AUC values Vs original AUC values.

**No-tottering extension:**
Figure 3.18, 3.19 and 3.20 respectively present the results of the no-tottering extension (3.7) of the size-based (3.1), branching-based (3.2), and until-N branching-based (3.6) kernels.

Several observations are consistent with the first dataset. We can likewise note that with the no-tottering extension, the introduction of tree-patterns is systematically beneficial in both kernels. Moreover, at a given order, optimal results observed with the two kernels are similar, and the corresponding $\lambda$ value is smaller with the branching-based kernel. Finally, the no-tottering extension limits the combinatorial explosion of the kernels computation.

There is however a striking difference because results are optimal here for patterns of order 3, patterns of order 2 rank second, and the results gradually decrease for orders greater than 3. This behavior is exactly opposite to the one we observed with the first dataset, where results gradually increased with the order of the patterns and were optimal for patterns of order 8. This therefore tends to confirm that distinct features are to be detected within the two datasets, and can be explained by the fact that the compounds are structurally similar in the first dataset, and different (or *noncongeneric*) in the second one. Indeed, while the kernel needs to detect subtle differences between the compounds of the first dataset, it must identify regular patterns within the second one, and it is not surprising that discriminating patterns are shorter in this case. This observation supports the intuition that the choice of the order of the patterns should to be related to (or learned from) the dataset itself, as suggested in section 3.3.1. Finally, we note that the best AUC value is around 84 % (corresponding to a relative improvement of 7% over the corresponding walk-based kernel), and is therefore similar to that obtained with the original formulation of the kernel. Nevertheless, we

observe from the curves on the right-hand side that contrary to the first dataset, the no-tottering extension has a limited overall impact. This is due to the surprising fact that here, the no-tottering extension does not seem to be beneficial by itself, since we can note that it systematically degrades the performance of the corresponding walk-based kernels, obtained for $\lambda = 0$. As a result, even though the introduction of tree-patterns is beneficial in both cases, better performances can be obtained here if we consider tottering tree-patterns. Once again this behavior is opposite to that of the first dataset. This might be explained as well by the fact that, contrary to the first dataset, the molecules considered here are structurally different, and as a result, tottering can help finding common features between these noncongeneric compounds.

Concerning the no-tottering extension (3.7) of the until-N branching-based kernel (3.6), results presented in Figure 3.20 are not clear. Indeed, in that case, the introduction of the tree-patterns only improves the results for patterns of limited order, and for patterns of order greater than 4, results systematically decrease. We can however note the interesting point that optimal results obtained for patterns of order 5 to 10 converge to a global optimal value between 85 and 86 %. This therefore tends to confirm that in the no-tottering case, the until-N extension can help solving the problem of pattern order selection by considering a maximal pattern order large enough (here, $h > 4$). Nevertheless, the striking difference with the results obtained with the first dataset is that in this case, when $h > 4$, the introduction of tree-patterns could not further improve the results obtained by the until-N walk-based kernel, that constitute the overall best performance we could observe for this dataset.

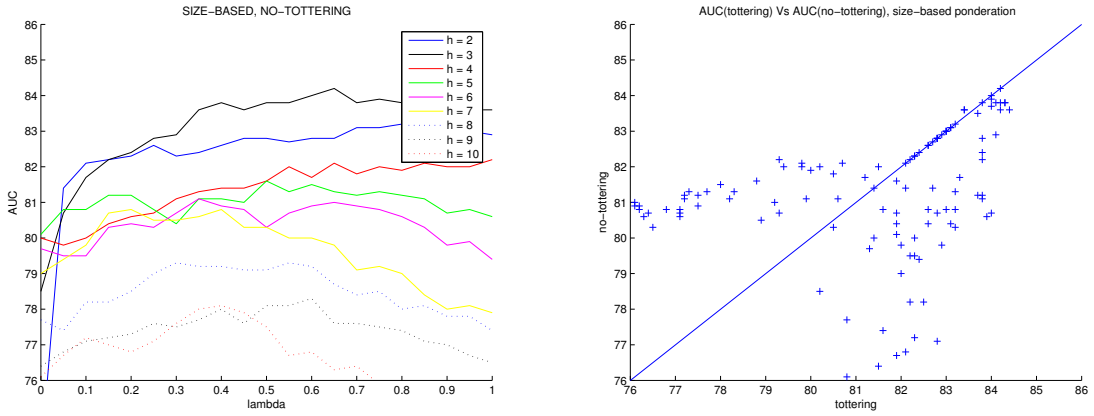


Figure 3.18: Second dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the size-based kernel (3.1). Right: no-tottering AUC values Vs original values.

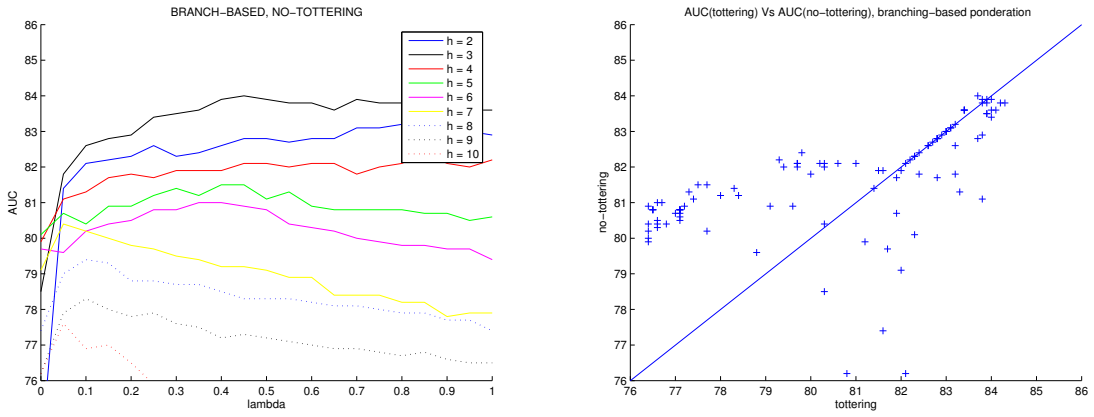


Figure 3.19: Second dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the branching-based kernel (3.2). Right: no-tottering AUC values Vs original values.

## 3.7 Discussion and conclusion

This paper introduces a family of graph kernels based on the detection of common tree patterns in the graphs. In a first step, we revisited an initial formulation presented in Ramon and Gärtner (2003), from which we derived two kernels with explicit feature spaces and inner products. A parameter $\lambda$ enters their definition and makes it possible to control the complexity of the features characterizing the graphs. At the extreme, admissible tree-patterns consist of linear chains of graph vertices, and the kernels resume to a classical graph kernel based on the detection of common walks (Gärtner et al., 2003). Walk-based graph kernels are therefore generalized to a wider class of kernels defined by features of increasing levels of complexity. In a second step we introduced two modular extensions to this initial formulation. On the one hand, the set of trees initially indexing the feature space is enriched by the set of their subtrees with an *until-N* extension, leading to a wider and more general feature space. On the other hand, a *no-tottering* extension prevents spurious tree-patterns to be detected, based on the notion of "tottering" initially introduced in the context of walk-based graph kernels (Mahé et al., 2005).

In the context of chemical applications, experiments on two toxicity datasets demonstrate that the tree-pattern graph kernels under their initial formulation improve over their walk-based counterpart. However, while a significant improvement could be observed for relatively small patterns, experiments revealed the difficulty to handle high order patterns. This is due to the fact that the number of tree-patterns detected in the graphs increases exponentially with their depth, which leads to a combinatorial explosion of the kernels computation for large patterns. For this reason, the until-N extension showed to be useless in this context: patterns of a given order are drowned within the flood of patterns of greater order, and the two kernel formulations turned

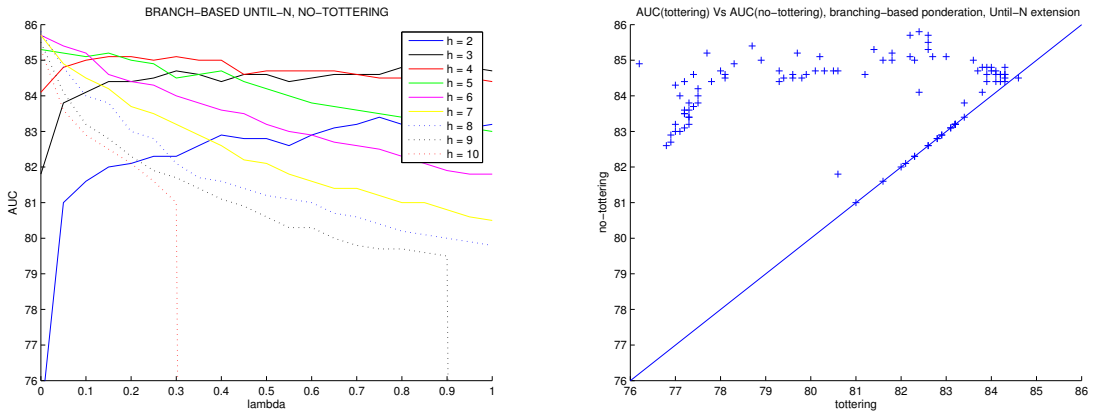


Figure 3.20: Second dataset. Left: evolution of the AUC with respect to $\lambda$ at different orders $h$ for the no-tottering extension (3.7) of the until-N branching-based kernel (3.6). Right: no-tottering AUC values Vs original values.

out to be equivalent. With the elimination of artificial tree-patterns, the no-tottering extension limits this combinatorial explosion, and patterns of higher order can be considered in the kernel. This was in particular beneficial to the first dataset where optimal results were obtained with high-order no-tottering patterns. Nevertheless, we notice that this extension is not always beneficial, and that in some cases, artificial common patterns due to the tottering phenomenon can help detecting molecular similarity. This is in particular the case for the second dataset, and can be explained by the fact that, in opposition to the first dataset, it consists of structurally different compounds. The combination of the two extensions led to mixed results. For the first dataset, we observe that the introduction of tree-patterns in this context could now improve over their walk-based counterparts for any maximum pattern order. This suggests that the limitation of the combinatorial explosion offered by the no-tottering extension makes it possible to combine patterns of different order in the kernel. However, albeit close, optimal results with the until-N extension could not come up with the optimal results that were obtained with no-tottering patterns of a given order. This suggests that very precise patterns were to be detected, and that their discriminative power is reduced by the addition of other, less predictive, patterns. For the second dataset, the combination of the two extensions led to optimal results. In that case however, the introduction of tree-patterns was not always beneficial and these optimal results were obtained by until-N, no-tottering walk-kernels. Finally, we can note that, when the maximum order of the patterns considered is large enough, results obtained with the until-N extension and no-tottering patterns tend to converge to a global optimum which is close, or equal to, to the overall best performance observed in both datasets.

Among the possible extensions to our work, we note that it might be relevant in the context of chemical applications to incorporate chemical knowledge in the graph representation of the molecules. For instance, it is well known that physico-chemical

properties of atoms are related to their position in the molecule, and as a first step in this direction, an enrichment of atom labels by their Morgan indices led to promising results in the context of walk-based kernels (Mahé et al., 2005). However, this particular approach is likely to have a lesser impact in this context, because the information encoded by the Morgan indices is at some extend already incorporated in the tree-patterns. Alternatively, we note that the kernel implementation could easily be extended in order to introduce a flexible matching between tree-patterns based on measures of similarity between pairs of vertices and edges, following for instance the construction of the marginalized kernel between labeled graphs (Kashima et al., 2004). Such an extension would induce an increase in the cost of computing the kernel, but is likely to make sense for chemical applications, where atoms of different types can exhibit similar properties.

# Appendix

## Proof of Propositions 13 and 14

In Propositions 13 and 14, we want to prove that for the graphs $G_1$ and $G_2$

$$\sum_{t\in\mathcal{B}_h} w(t)\psi_t(G_1)\psi_t(G_2) = \alpha(h) \sum_{u\in\mathcal{V}_{G_1}} \sum_{v\in\mathcal{V}_{G_2}} k_h(u,v), \tag{3.8}$$

where in Proposition 13, $\alpha(h) = \lambda^{-h}$ and $w(t) = \lambda^{|t|-h}$, while in Proposition 14, $\alpha(h) = 1$ and $w(t) = \lambda^{\text{branch}(t)}$.

From Definition 8 we have $\psi_t(G) = \sum_{u\in\mathcal{V}_G} \psi_t^{(u)}(G)$. As a result,

$$\sum_{t\in\mathcal{B}_h} w(t)\psi_t(G_1)\psi_t(G_2) = \sum_{u\in\mathcal{V}_{G_1}} \sum_{v\in\mathcal{V}_{G_2}} \Big( \sum_{t\in\mathcal{B}_h} w(t)\psi_t^{(u)}(G_1)\psi_t^{(v)}(G_2) \Big),$$

and in order to prove (3.8) we just need to prove

$$\sum_{t\in\mathcal{B}_h} w(t)\psi_t^{(u)}(G_1)\psi_t^{(v)}(G_2) = \alpha(h)k_h(u,v). \tag{3.9}$$

### Proof of Proposition 13

In order to prove Proposition 13, it follows from (3.9) that we just need to prove that

$$\frac{1}{\lambda^h} k_h(u,v) = \sum_{t\in\mathcal{B}_h} \lambda^{|t|-h} \psi_t^{(u)}(G_1)\psi_t^{(v)}(G_2)\,,$$

or equivalently:

$$k_h(u,v) = \sum_{t\in\mathcal{B}_h} \lambda^{|t|} \psi_t^{(u)}(G_1)\psi_t^{(v)}(G_2)\,, \tag{3.10}$$

where $k_h$ is defined recursively by $k_1(u,v) = \lambda \mathbf{1}(l(u) = l(v))$ and for $h > 1$:

$$k_h(u,v) = \lambda \mathbf{1}(l(u) = l(v)) \sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} k_{h-1}(u',v') \,. \tag{3.11}$$

We prove (3.10) by induction on $h$. The case $h = 1$ is rather trivial. Indeed, a tree of depth one is just a single node, and $\psi_t^{(u)}(G_1)$ is therefore equal to 1 if $l(u) = l(r(t))$, 0 otherwise. It follows that

$$\begin{aligned} \sum_{t \in \mathcal{B}_1} \lambda^{|t|} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2) &= \sum_{t \in \mathcal{B}_1} \lambda \mathbf{1}(l(r(t)) = l(u)) \mathbf{1}(l(r(t)) = l(v)) \\ &= \lambda \mathbf{1}(l(u) = l(v)), \end{aligned}$$

which corresponds to $k_1(u,v)$.

Let us now assume that (3.10) is true at order $h-1$, and let us prove that it is then also true at order $h > 1$. Combining the recursive definition of $k_h$ (3.11) with the induction hypothesis (3.10) at level $h-1$ we first obtain:

$$k_h(u,v) = \lambda \mathbf{1}(l(u) = l(v)) \sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} \sum_{t' \in \mathcal{B}_{h-1}} \lambda^{|t'|} \psi_{t'}^{(u')}(G_1) \psi_{t'}^{(v')}(G_2) \,. \tag{3.12}$$

Second, for any graph $G$, let us denote by $\mathcal{P}_n^{(u)}(G)$ the set of balanced tree-patterns of order $n$ rooted in $u \in \mathcal{V}_G$, and for any tree-pattern $p \in \mathcal{P}_n^{(u)}(G)$ let $t(p) \in \mathcal{B}_n$ denote the corresponding tree. With these notations we can rewrite, for any $n \geq 1$ and $(u,v) \in G_1 \times G_2$:

$$\sum_{t \in \mathcal{B}_n} \lambda^{|t|} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2) = \sum_{p_1 \in \mathcal{P}_n^{(u)}(G_1)} \sum_{p_2 \in \mathcal{P}_n^{(v)}(G_2)} \lambda^{|t(p_1)|} \mathbf{1}(t(p_1) = t(p_2)). \tag{3.13}$$

Indeed both sides of this equation count the number of pairs of similar tree-patterns rooted in $u$ and $v$. Plugging (3.13) into (3.12) we get:

$$k_h(u,v) = \lambda \mathbf{1}(l(u) = l(v)) \sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} \sum_{p_1 \in \mathcal{P}_{h-1}^{(u')}(G_1)} \sum_{p_2 \in \mathcal{P}_{h-1}^{(v')}(G_2)} \lambda^{|t(p_1)|} \mathbf{1}(t(p_1) = t(p_2)) \,. \tag{3.14}$$

Now we use the fact that any tree-pattern $p$ of order $h$ can be uniquely decomposed into a tree-pattern $p'$ of order 2 and a set of tree-patterns of order $h-1$ rooted at the leaves of $p'$. We note that matching two tree-patterns is equivalent to matching the tree-patterns in their decomposition, and that the sets of leaves of tree-patterns of order 2 rooted respectively in $u$ and $v$ matching each other are exactly given by $\mathcal{M}(u,v)$. In other words, (3.14) performs a summation over pairs of matching tree-patterns of depth $h$, rooted respectively in $u$ and $v$: the corresponding pairs of patterns of order 2 are implicitly matched by the summation over $\mathcal{M}(u,v)$ and the condition

$\mathbf{1}(l(u) = l(v))$, and the subsequent pairs of patterns $(p_1, p_2)$ of order $h-1$ are matched by the product of conditions $\mathbf{1}(t(p_1) = t(p_2))$.

The tree-pattern $p_1$ in $G_1$ of such a matching pair of tree-patterns of order $h$ rooted in $(u, v)$ decomposes as a pattern of depth 2 rooted in $u$ with leaves in some $R \in \mathcal{M}(u, v)$, and a set of patterns $p_1(u')$ of depth $h-1$ rooted in the leaves $u' \in R$. By (3.14), to each such matching pair is associated the weight $\lambda \times \prod_{(u',v') \in R} \lambda^{|t(p1(u'))|}$, which is exactly equal to $\lambda^{|t(p_1)|}$ since we obviously have $|t(p_1)| = 1 + \sum_{(u',v') \in R} |t(p_1(u'))|$. As a result, (3.14) can be rewritten as:

$$k_h(u, v) = \sum_{p_1 \in \mathcal{P}_h^{(u)}(G_1)} \sum_{p_2 \in \mathcal{P}_h^{(v)}(G_2)} \lambda^{|t(p_1)|} \mathbf{1}(t(p_1) = t(p_2)),$$

which combined with (3.13) proves (3.10). ■

**Proof of Proposition 14**

The proof of Proposition 14 is a straightforward variant of the proof of Proposition 13. By (3.9) we need to show that

$$k_h(u, v) = \sum_{t \in \mathcal{B}_h} \lambda^{\text{branch}(t)} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2)\,, \tag{3.15}$$

where $k_h$ is defined recursively by $k_1(u, v) = \mathbf{1}(l(u) = l(v))$ and for $h > 1$:

$$k_h(u, v) = \frac{\mathbf{1}(l(u) = l(v))}{\lambda} \sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} \lambda k_{h-1}(u', v')\,. \tag{3.16}$$

We proceed again by induction over $h$ to prove (3.15). The case $h = 1$ is easily done by checking, using an argument similar to that of the previous proof, that (3.15) is one if $l(u)$ and $l(v)$ are identical, zero otherwise, which corresponds to the definition of $k_1(u, v)$. If we assume that (3.15) is true at the level $h-1$, we can plug it in (3.16) to obtain:

$$k_h(u, v) = \frac{\mathbf{1}(l(u) = l(v))}{\lambda} \sum_{R \in \mathcal{M}(u,v)} \prod_{(u',v') \in R} \sum_{t' \in \mathcal{B}_{h-1}} \lambda^{1+\text{branch}(t')} \psi_{t'}^{(u')}(G_1) \psi_{t'}^{(v')}(G_2)\,. \tag{3.17}$$

We can then follow exactly the same line of proof as in the previous section and obtain the following equations

$$\sum_{t \in \mathcal{B}_n} \lambda^{\text{branch}(t)} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2) = \sum_{p_1 \in \mathcal{P}_n^{(u)}(G_1)} \sum_{p_2 \in \mathcal{P}_n^{(v)}(G_2)} \lambda^{\text{branch}(t(p_1))} \mathbf{1}(t(p_1) = t(p_2))\,, \tag{3.18}$$

and

$$k_h(u,v) = \frac{\mathbf{1}(l(u)=l(v))}{\lambda} \sum_{R\in\mathcal{M}(u,v)} \prod_{(u',v')\in R} \sum_{p_1\in\mathcal{P}_{h-1}^{(u')}(G_1)} \sum_{p_2\in\mathcal{P}_{h-1}^{(v')}(G_2)} \lambda^{1+\text{branch}(t(p_1))}\mathbf{1}(t(p_1)=t(p_2))\,, \tag{3.19}$$

that correspond respectively to (3.13) and (3.14). The only difference with the previous proof is in the exponent of $\lambda$ to form the weight of a matching pair of tree-patterns. By analogy with the previous proof, we consider the tree-pattern $p_1$ in $G_1$ of a pair of matching tree-patterns of depth $h$ rooted in $(u,v)$, that decomposes as a pattern of depth 2 rooted in $u$ with leaves in some $R\in\mathcal{M}(u,v)$, and a set of patterns $p_1(u')$ of depth $h-1$ rooted in the leaves $u'\in R$. By (3.19), to each such matching pair is associated the weight $\frac{1}{\lambda}\prod_{(u',v')\in R}\lambda^{1+\text{branch}(t(p_1(u')))} = \lambda^{-1+\sum_{(u',v')\in R}1+\text{branch}(t(p_1(u')))}$. We observe that the number of leaves of a tree $t$, that we note leaves$(t)$, is equal to $1+\text{branch}(t)$. The weight associated to the above pair of matching tree-patterns can therefore be written as $\lambda^{-1+\sum_{(u',v')\in R}\text{leaves}(t(p_1(u')))}$. Finally, because the number of leaves of the tree-pattern $p_1$ is equal to the sum of the leaves of the patterns $p_1(u')$, it follows that this expression is equal to $\lambda^{-1+\text{leaves}(t(p_1))} = \lambda^{\text{branch}(t(p_1))}$. As a result, we can write (3.19) as

$$k_h(u,v) = \sum_{p_1\in\mathcal{P}_h^{(u)}(G_1)} \sum_{p_2\in\mathcal{P}_h^{(v)}(G_2)} \lambda^{\text{branch}(t(p_1))}\mathbf{1}(t(p_1)=t(p_2)),$$

which, combined with (3.18), concludes the proof. ■

## Proof of Proposition 15

The proof presented in this section is very similar to the proofs of Propositions 13 and 14. Based on the observations made in the beginning of Appendix 3.7, it follows from (3.9) that in order to prove Proposition 15, we just need to prove that

$$k_h(u,v) = \sum_{t\in\mathcal{T}_h} \lambda^{\text{branch}(t)}\psi_t^{(u)}(G_1)\psi_t^{(v)}(G_2)\,, \tag{3.20}$$

where $k_h$ is defined recursively by $k_1(u,v)=\mathbf{1}(l(u)=l(v))$ and for $h>1$

$$k_h(u,v) = \mathbf{1}(l(u)=l(v))\Big(1+\sum_{R\in\mathcal{M}(u,v)}\frac{1}{\lambda}\prod_{(u',v')\in R}\lambda k_{h-1}(u',v')\Big). \tag{3.21}$$

We proceed again by induction over $h$ to prove (3.20). The case $h=1$ directly follows from the proof of Proposition 14. If we assume that (3.20) is true at the level $h-1$, we can plug it in (3.21) to obtain:

$$k_h(u,v) = \mathbf{1}(l(u)=l(v))\Big(1+\sum_{R\in\mathcal{M}(u,v)}\frac{1}{\lambda}\prod_{(u',v')\in R}\sum_{t'\in\mathcal{T}_{h-1}}\lambda^{1+\text{branch}(t')}\psi_{t'}^{(u')}(G_1)\psi_{t'}^{(v')}(G_2)\Big). \tag{3.22}$$

By analogy with the construction of the previous proof, for any graph $G$, let us denote by $\mathcal{P}_n^{(u)}(G)$ the set of tree-patterns of depth 1 to $n$ rooted in $u \in \mathcal{V}_G$, and for any tree-pattern $p \in \mathcal{P}_n^{(u)}(G)$ let $t(p) \in \mathcal{T}_n$ denote the corresponding tree. Note that $\mathcal{P}_n^{(u)}(G)$ corresponds here to general tree-patterns of depth 1 to $n$, in opposition to the balanced-tree patterns of order $n$ involved in the previous proofs. With these notations we obtain similarly, for any $n \geq 1$ and $(u, v) \in G_1 \times G_2$:

$$\sum_{t \in \mathcal{T}_n} \lambda^{\text{branch}(t)} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2) = \sum_{p_1 \in \mathcal{P}_n^{(u)}(G_1)} \sum_{p_2 \in \mathcal{P}_n^{(v)}(G_2)} \lambda^{\text{branch}(t(p_1))} \mathbf{1}(t(p_1) = t(p_2)), \tag{3.23}$$

and, plugging (3.23) into (3.22), we get:

$$\begin{aligned} k_h(u, v) =& \mathbf{1}(l(u) = l(v)) \\ & \times \Big(1 + \sum_{R \in \mathcal{M}(u,v)} \frac{1}{\lambda} \prod_{(u', v') \in R} \sum_{p_1 \in \mathcal{P}_{h-1}^{(u')}(G_1)} \sum_{p_2 \in \mathcal{P}_{h-1}^{(v')}(G_2)} \lambda^{1+\text{branch}(t(p_1))} \mathbf{1}(t(p_1) = t(p_2))\Big), \end{aligned} \tag{3.24}$$

which can be further decomposed into:

$$\begin{aligned} k_h(u, v) =& \mathbf{1}(l(u) = l(v)) \\ & + \frac{\mathbf{1}(l(u) = l(v))}{\lambda} \sum_{R \in \mathcal{M}(u,v)} \prod_{(u', v') \in R} \sum_{p_1 \in \mathcal{P}_{h-1}^{(u')}(G_1)} \sum_{p_2 \in \mathcal{P}_{h-1}^{(v')}(G_2)} \lambda^{1+\text{branch}(t(p_1))} \mathbf{1}(t(p_1) = t(p_2)). \end{aligned} \tag{3.25}$$

The second part of the right member of (3.25) matches pairs of tree-patterns of depth 2 to $n$ rooted in $(u, v)$. It follows directly from the proof of Proposition 14 that such a pair $(p_1, p_2)$ of matching tree-patterns is weighted by $\lambda^{\text{branch}(t(p_1))}$. The first part of the right member of (3.25) matches the trivial pair of tree-patterns of depth 1 rooted in $(u, v)$ consisting of the single nodes $(u, v)$. The corresponding tree has a zero branching cardinality, and we can therefore write

$$\mathbf{1}(l(u) = l(v)) = \sum_{t \in \mathcal{T}_1} \lambda^{\text{branch}(t)} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2).$$

Taken together, these two arguments show that (3.25) can be written as

$$k_h(u, v) = \sum_{t \in \mathcal{T}_h} \lambda^{\text{branch}(t)} \psi_t^{(u)}(G_1) \psi_t^{(v)}(G_2),$$

which concludes the proof. ■

## Proof of Proposition 19

The proof is derived from results presented in Mahé et al. (2005). The sets of walks and no-tottering walks of the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ are respectively defined by $\mathcal{W}(G) = \bigcup_{n=0}^{\infty} \mathcal{W}_n(G)$ and $\mathcal{W}^{NT}(G) = \bigcup_{n=0}^{\infty} \mathcal{W}_n^{NT}(G)$, where

$$\mathcal{W}_n(G) = \{(v_0, \ldots, v_n) \in \mathcal{V}_G^{n+1} : (v_i, v_{i+1}) \in \mathcal{E}_G,\ 0 \leq i \leq n-1\}$$

is the set of walks of length $n$ defined is Section 3.3.1, and

$$\mathcal{W}_n^{NT}(G) = \{(v_0, \ldots, v_n) \in \mathcal{W}_n(G) :\ v_i \neq v_{i+2},\ 0 \leq i \leq n-2\}$$

is the set of no-tottering walks of length $n$ defined in Mahé et al. (2005). We start by stating the following lemma.

**Lemma 20.** *A tree-pattern $p$ of the graph $G$ associated to the tree $t$ is no tottering if, and only if, any walk of $G$ defined as a succession of vertices of $p$ corresponding to nodes of $t$ forming a path from its root to one of its leaves is no-tottering.*

*Proof.* [Lemma 20] According to Definition 16, let $(v_1, \ldots, v_{|t|}) \in \mathcal{V}_G^{|t|}$ be a no-tottering tree pattern of the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ corresponding to the tree $t = (\mathcal{V}_t, \mathcal{E}_t)$, where $\mathcal{V}_t = (n_1, \ldots, n_{|t|})$. Let $(n_{i_0}, \ldots, n_{i_k}) \in \mathcal{V}_t^{k+1}$ be a path from the root of $t$ to one of its leaves. By Definition 7, it is clear that $(v_{i_0}, \ldots, v_{i_k}) \in \mathcal{W}(G)$. Moreover, by the definition of paths we have $(n_{i_m}, n_{i_{m+1}}), (n_{i_{m+1}}, n_{i_{m+2}}) \in \mathcal{E}_t$ for $0 \leq m \leq k-2$. By Definition 16, this implies that $v_{i_m} \neq v_{i_{m+2}}$ for $0 \leq m \leq k-2$, meaning that $(v_{i_0}, \ldots, v_{i_k}) \in \mathcal{W}^{NT}(G)$.

Conversely, let $p \in \mathcal{V}_G^{|t|}$ be a tree-pattern of the graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ corresponding to the tree $t = (\mathcal{V}_t, \mathcal{E}_t)$. Consider the set of walks of $G$ defined as successions of vertices of $p$ associated to nodes of $t$ forming paths from its root to its leaves. If these walks are not tottering, it is clear from Definition 16 that the tree-pattern itself is not tottering. ■

We can now state the proof of Proposition 19.

*Proof.* [Proposition 19] If, according to Definition 18, we let $G'$ be the transformed graph of $G$, Mahé et al. (2005) showed that there is a bijection between $\mathcal{W}^{NT}(G)$ and the set of walks of $G'$ starting in a vertex corresponding to a vertex of $G$, which can be formally defined as

$$\mathcal{W}^{\{V_G\}}(G') = \{(v_0, \ldots, v_n) \in \mathcal{W}(G') :\ v_0 \in \{V_G\}, n \in \mathbb{N}\},$$

if we let $V_G \subset \mathcal{V}_{G'}$ be the subset of $\mathcal{V}_{G'}$ that corresponds to $\mathcal{V}_G$. It follows from Lemma 20 that there is a bijection between the set of no-tottering tree-patterns of $G$ and the set of tree-patterns of $G'$ rooted in a vertex of $V_G$. Finally, Mahé et al. (2005) showed that a walk in $\mathcal{W}^{NT}(G)$ and its image in $\mathcal{W}^{\{V_G\}}(G')$ are identically labeled, which enables to count no-tottering labeled walks in $G$, by counting identically labeled walks in $G'$ starting in a vertex of $V_G$ . It follows that counting no-tottering tree-patterns in $G$ is equivalent to counting tree-patterns in $G'$ rooted in a vertex of $V_G$. As a result, we have $\psi_t^{NT}(G) = \psi_t^{\{V_G\}}(G')$, which concludes the proof. ■

Chapter 4

# The pharmacophore kernel

This work appeared in a slightly different form in the *Journal of Chemical Information and Modeling*, co-authored with Liva Ralaivola, Véronique Stoven and Jean-Philippe Vert (Mahé et al., 2006).

## Introduction

It is widely accepted that several drug-like properties can be efficiently deduced from the 2D structure of the molecule, that is, the description of a molecule as a set of atoms and their covalent bonds. For example, Lipinski's "rule of five" remains a widely used standard for the prediction of intestinal absorption (Lipinski et al., 2001), and the prediction of mutagenicity from 2D molecular fragments is an accurate state-of-the-art approach (King et al., 1996). In the case of target binding prediction, however, 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 (Böhm et al., 2003). For this reason, there has been a long history of research on the prediction of these interactions from the 3D representation of molecules, that is, their spatial conformation in the 3D space. If the 3D structure of the target is known, the strength of the interaction can be directly evaluated by docking techniques, that quantify the complementarity of the molecule to the target in terms of energy (Halperin et al., 2002; Kitchen et al., 2004). In the general case where the 3D structure of the target is unknown, however, the docking approach is not possible anymore and the modeler must adopt a ligand-based approach to create a predictive model from available data, typically a pool of molecules with known affinity to the target.

Most approaches to ligand-based virtual screening require to represent and compare 3D structures of molecules. The comparison of 3D structures can for example rely on optimal alignments in the 3D space (Lemmen and Lengauer, 2000), or on the comparison of features extracted from the structures (Xue and Bajorath, 2000). Fea-

tures of particular importance in this context are subsets of two to four atoms together with their relative spatial organization, also called *pharmacophores*. Discovering pharmacophores common to a set of known inhibitors to a drug target can be a powerful approach to the screening of other candidate molecules containing the pharmacophores, as well as a first step towards the understanding of the biological phenomenon involved (Holliday and Willett, 1997; Finn et al., 1998). Alternatively, pharmacophore fingerprints, that is, bitstrings representing a molecule by the pharmacophores it contains, has emerged as a potential approach to apply statistical learning methods for SAR, although sometimes with mixed results (Matter and Pötter, 1999; Brown and Martin, 1997; Bajorath, 2001).

We focus in this paper on an extension of the fingerprint representation of molecules for building SAR models with support vector machines (SVM). Although SVM can be trained from a vector or bitstring representation of molecules, they can also take advantage of a mathematical trick to only rely on a measure of similarity between molecules, known as *kernel*. This trick was for example used in Kashima et al. (2004) and Mahé et al. (2005) to build SAR models from a 2D fingerprint of molecules of virtually infinite length. Here we investigate the possibility to use this trick in the context of 3D SAR modeling. We propose a measure of similarity between 3D structures, which we call the *pharmacophore kernel*, based on the comparison of pharmacophores present in the structures. It satisfies the mathematical properties required to be a valid kernel and it therefore allows the use of SVM for model building. This kernel bears some similarity with pharmacophore fingerprint approaches, although it produces more general models. In fact, we show that a fast approximation of this kernel, based on pharmacophore fingerprints, leads to significantly lower performance on a benchmark dataset. The overall good performance of the approach on this benchmark supports its relevance as a potentially effective tool for 3D SAR modeling. We start this chapter by the definition of the pharmacophore kernel in Section 4.1. Its exact computation is presented in Section 4.2, followed by a discussion about the connection between the pharmacophore kernel and recently introduced graph kernels (Section 4.3), and the presentation of a fast approximation (Section 4.4). Experimental results on a benchmark dataset involving the detection of inhibitors of several drug targets are then presented in Section 4.5, followed by a short discussion.

## 4.1 Kernel definition

A *pharmacophore* is usually defined as a three-dimensional arrangement of atoms - or groups of atoms - responsible for the biological activity of a drug molecule (Güner, 2000). The present work focuses on *three-points pharmacophores*, composed of three atoms whose arrangement therefore forms a triangle in the 3D space (Figure 4.1). With a slight abuse we refer as pharmacophore below to *any* possible configuration of three atoms or classes of atoms arranged as a triangle and present in a molecule, representing therefore a *putative* configuration responsible for the biological property

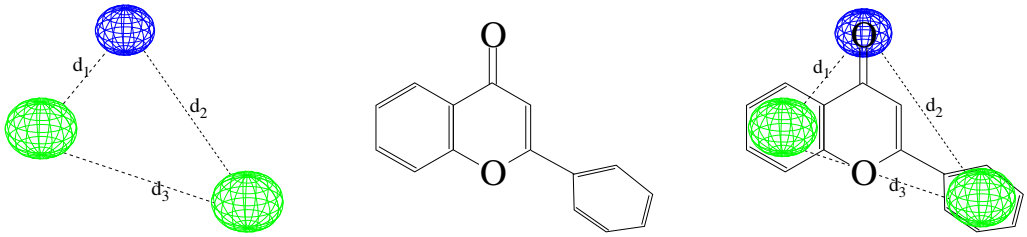


Figure 4.1: Left: a 3-points pharmacophore made of one hydrogen bond acceptor (topmost sphere) and two aromatic rings, with distances $d_1$, $d_2$ and $d_3$ between the features. Middle: the molecule of flavone. Right: match between flavone and the pharmacophore.

of interest.

Throughout this paper we represent the 3D structure of a molecule as a set of points in $\mathbb{R}^3$. These points correspond to the 3D coordinates of the atoms of the molecule (for a given arbitrary basis of the 3D Euclidean space), and they are labeled with some information related to the atoms. More formally, we define a molecule $m$ as

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

where $|\,m\,|$ is the number of atoms that compose the molecule and $\mathcal{L}$ denotes the set of atom labels. The label is meant to contain the relevant information to characterize a pharmacophore based on atoms. It might for instance be defined by the type of atom (C, N, O, ...) and/or various physicochemical atomic properties (e.g., partial charge). The three-points pharmacophores considered in this work correspond to triplets of distinct atoms of the molecules. The set of pharmacophores of the molecule $m$ can therefore be formally defined as:

$$\mathcal{P}(m) = \left\{(p_1, p_2, p_3) \in m^3, p_1 \neq p_2, p_1 \neq p_3, p_2 \neq p_3\right\} \ . \tag{4.1}$$

More generally, the set of all possible pharmacophores is naturally defined as $\mathcal{P} = (\mathbb{R}^3 \times \mathcal{L})^3$, to ensure the inclusion $\mathcal{P}(m) \subset \mathcal{P}$. We can now define a general family of kernels for molecules based on their pharmacophore content:

**Definition 21.** *For any positive definite kernel for pharmacophores $K_{\mathcal{P}} : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$, we define a corresponding* pharmacophore kernel *for any pair of molecules $m$ and $m'$ by :*

$$K(m, m') := \sum_{p \in \mathcal{P}(m)} \sum_{p' \in \mathcal{P}(m')} K_{\mathcal{P}}(p, p') \ , \tag{4.2}$$

*with the convention that $K(m, m') = 0$ if either $\mathcal{P}(m)$ or $\mathcal{P}(m')$ is empty.*

The fact that the pharmacophore kernel defined in (4.2) is a valid positive definite kernel on the set of molecules, as soon as $K_{\mathcal{P}}$ is itself a valid positive definite kernel on the set of pharmacophores, is a classical result (see e.g., Haussler, 1999, Lemma 1).

The problem of constructing a pharmacophore kernel for molecules therefore boils down to the simpler problem of defining a kernel between pharmacophores. A chemically relevant measure of similarity between pharmacophores should obviously quantify at least two features: first, similar pharmacophores should be made of similar atoms (where the notion of similarity can for instance be based on atom types or property(ies), and more generally on pharmacophoric types), and second, the atoms should have similar relative positions in the 3D space. It is therefore natural to study kernels for pharmacophores that decompose as follows:

$$K_{\mathcal{P}}(p, p') = K_I(p, p') \times K_S(p, p') \ , \tag{4.3}$$

where $K_I$ is a kernel function assessing the similarity between the triplets of basis atoms of the pharmacophores (their so-called *intrinsic* similarity), and $K_S$ is a kernel function introduced to quantify their *spatial* similarity.

We can furthermore investigate intrinsic and spatial kernels that factorize themselves as products of more basic kernels between atoms and pairwise distances, respectively. Triplets of atoms are indeed globally similar if the three corresponding pairs of atoms are simultaneously similar, and triangles are similar if the lengths of their edges are pairwise similar. For any pair of pharmacophores $p = ((x_1, l_1), (x_2, l_2), (x_3, l_3))$ and $p' = ((x'_1, l'_1), (x'_2, l'_2), (x'_3, l'_3))$, this suggests to define kernels as follows:

$$K_I(p, p') = \prod_{i=1}^{3} K_{Feat}(l_i, l'_i) \ , \tag{4.4}$$

$$K_S(p, p') = \prod_{i=1}^{3} K_{Dist}\left(\| x_i - x_{i+1} \|, \| x'_i - x'_{i+1} \|\right) \ , \tag{4.5}$$

where $\| . \|$ denotes the Euclidean distance, the index $i+1$ is taken modulo 3, and $K_{Feat}$ and $K_{Dist}$ are kernels functions introduced to compare pairs of labels from $\mathcal{L}$, and pairs of distances, respectively. It suffices now to define the kernels $K_{Feat}$ on $\mathcal{L} \times \mathcal{L}$ and $K_{Dist}$ on $\mathbb{R} \times \mathbb{R}$ in order to obtain, by (4.2), (4.3), (4.4) and (4.5), a pharmacophore kernel for molecules. The first one compares the atom labels, while the second compares the distances between atoms in the pharmacophores. Intuitively they define the basic notions of similarity involved in the pharmacophore comparison, which in turns defines the overall similarity between molecules.

Note from the definition in equation (4.1) that, because of permutations, every distinct triplet of atoms of the molecule $m$ gives rise to six pharmacophores in $\mathcal{P}(m)$. In the general case, these six pharmacophores are considered as different in the pharmacophore kernel. However, because of the definition of the notion of similarity between pharmacophores, some of these six pharmacophores will be seen as identical if the triplet of atoms is made of atoms in the same type. This phenomenon is even emphasized if the triplet of atoms exhibits some kind of spatial symmetry. For example, the six pharmacophores associated to a triplet of identical atoms arranged as an equilateral triangle are identical.

The kernel we use for $K_{Dist}$ is the Gaussian radial basis function (RBF) kernel, known to be a safe default choice for SVM working on real numbers or vectors (Schölkopf and Smola, 2002):

$$K_{Dist}^{RBF}(x,y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right) , \tag{4.6}$$

where $\sigma > 0$ is the bandwith parameter that will be optimized as part of the training of the classifier (see Section 4.5.2). Various kernels $K_{Feat}$ between labels can be chosen depending on the atom labels definition. These labels belonging in principle to a finite set of possible labels, e.g., the set of atom types with their charges ($C$, $C^+$, $C^-$, N, ...), the following *Dirac kernel* is a natural default choice to compare a pair of atom labels $l, l' \in \mathcal{L}$ :

$$K_{Feat}^{Dirac}(l,l') = \begin{cases} 1 & \text{if } l = l' , \\ 0 & \text{otherwise .} \end{cases} \tag{4.7}$$

Alternatively, it might be relevant for pharmacophore definition to compare atoms not only on the basis of their types and partial charges, but also in terms of other physicochemical parameters such as their size, polarity and electronegativity. Formally, a physicochemical parameter for an atom with label $l$ is a real number $f(l)$. In that case, the Gaussian RBF kernel (4.6) could be applied directly to the parameter values to compare labels. We discuss this issue in section 4.6. Note finally that the Gaussian (4.6) and Dirac (4.7) kernels are known to be definite positive (Schölkopf and Smola, 2002), and it follows from the closure properties of the family of kernel functions, that the kernel between pharmacophores $K_{\mathcal{P}}$ is valid for these choices of the kernels $K_{Feat}$ and $K_{Dist}$.

## 4.2 Kernel computation

We are now left with the task of computing the pharmacophore kernel (4.2) for a particular choice of feature and distance kernels $K_{Feat}$ and $K_{Dist}$. In this section we provide a simple analytical formula for this computation.

For any pair of molecules $m = \{(x_i, l_i) \in \mathbb{R}^3 \times \mathcal{L}\}_{i=1,\dots,|m|}$ and $m' = \{(x'_i, l'_i) \in \mathbb{R}^3 \times \mathcal{L}\}_{i=1,\dots,|m'|}$, let us define a square matrix $M$ of size $n = |m| \times |m'|$, whose dimensions are indexed by the Cartesian product of $m$ and $m'$. In other words, to each index $i \in [1,n]$ corresponds a unique couple of indices $(i_1, i_2) \in [1,|m|] \times [1,|m'|]$, and to each dimension of the matrix $M$ corresponds a distinct pair of points taken from the molecules $m$ and $m'$. Denoting by $\mathbf{1}(.)$ the indicator function equal to one if its argument is true, zero otherwise, the entries of $M$ are defined by:

$$\begin{aligned} M[i,j] &= M[(i_1,i_2),(j_1,j_2)] \\ &= K_{Feat}\big(l_{i_1}, l'_{i_2}\big) \times K_{Dist}\big(||x_{i_1} - x_{j_1}||, ||x'_{i_2} - x'_{j_2}||\big) \\ &\quad \times \mathbf{1}(i_1 \neq j_1) \times \mathbf{1}(i_2 \neq j_2) . \end{aligned} \tag{4.8}$$

The value of the pharmacophore kernel between $m$ and $m'$ can now be deduced from the matrix $M$ by the following result:

**Proposition 22.** *The pharmacophore kernel (4.2) between a pair of molecules $m$ and $m'$ is equal to:*

$$K(m, m') = \mathit{trace}(M^3),$$

*where $M$ is the square matrix of dimension $|\, m \,| \times |\, m' \,|$ constructed from $m$ and $m'$ by (4.8).*

*Proof.* Developing the matrix products involved in the expression of $M^3$ we get

$$\mathrm{trace}(M^3) = \sum_{i,j,k=1}^{n} M[i,j]M[j,k]M[k,i] \ ,$$

where $n = |m| \times |m'|$ is the size of $M$. Using the fact that the indices of $M$ ranges over the Cartesian product of the set of indices $[1, |m|]$ and $[1, |m'|]$, we can rewrite this expression as :

$$\mathrm{trace}(M^3) = \sum_{i_1,j_1,k_1=1}^{|m|} \sum_{i_2,j_2,k_2=1}^{|m'|} M[(i_1,i_2),(j_1,j_2)]M[(j_1,j_2),(k_1,k_2)]M[(k_1,k_2),(i_1,i_2)] \ .$$

Substituting with the definition of $M$ given in (4.8), we obtain :

$$\begin{aligned}
\mathrm{trace}(M^3) &= \sum_{i_1,j_1,k_1=1}^{|m|} \sum_{i_2,j_2,k_2=1}^{|m'|} \mathbf{1}\left(i_1 \neq j_1\right) \mathbf{1}\left(j_1 \neq k_1\right) \mathbf{1}\left(k_1 \neq i_1\right) \times \mathbf{1}\left(i_2 \neq j_2\right) \mathbf{1}\left(j_2 \neq k_2\right) \mathbf{1}\left(k_2 \neq i_2\right) \\
&\qquad \times K_{Feat}\left(l_{i_1}, l'_{i_2}\right) \times K_{Dist}\left(||x_{j_1} - x_{i_1}||, ||x'_{j_2} - x'_{i_2}||\right) \\
&\qquad \times K_{Feat}\left(l_{j_1}, l'_{j_2}\right) \times K_{Dist}\left(||x_{k_1} - x_{j_1}||, ||x'_{k_2} - x'_{j_2}||\right) \\
&\qquad \times K_{Feat}\left(l_{k_1}, l'_{k_2}\right) \times K_{Dist}\left(||x_{i_1} - x_{k_1}||, ||x'_{i_2} - x'_{k_2}||\right) \\
&= \sum_{i_1,j_1,k_1=1}^{|m|} \sum_{i_2,j_2,k_2=1}^{|m'|} \mathbf{1}\left(i_1 \neq j_1 \neq k_1\right) \times \mathbf{1}\left(i_2 \neq j_2 \neq k_2\right) \\
&\qquad \times K_{\mathcal{P}}\left(\left((x_{i_1}, l_{i_1}), (x_{j_1}, l_{j_1}), (x_{k_1}, l_{k_1})\right), \left((x'_{i_2}, l'_{i_2}), (x'_{j_2}, l'_{j_2}), (x'_{k_2}, l'_{k_2})\right)\right) \\
&= \sum_{\substack{i_1,j_1,k_1=1,\\ i_1 \neq j_1 \neq k_1}}^{|m|} \sum_{\substack{i_2,j_2,k_2=1,\\ i_2 \neq j_2 \neq k_2}}^{|m'|} K_{\mathcal{P}}\left(\left((x_{i_1}, l_{i_1}), (x_{j_1}, l_{j_1}), (x_{k_1}, l_{k_1})\right), \left((x'_{i_2}, l'_{i_2}), (x'_{j_2}, l'_{j_2}), (x'_{k_2}, l'_{k_2})\right)\right) \\
&= \sum_{p \in \mathcal{P}(m)} \sum_{p' \in \mathcal{P}(m')} K_{\mathcal{P}}(p, p') \\
&= K(m, m') \ .
\end{aligned}$$

■

If we let $u$ be the cost of evaluating the basis kernels $K_{Feat}$ and $K_{Dist}$, and consider that the cost of the addition and product operations is a small constant, the complexity of the kernel between pharmacophores $K_{\mathcal{P}}$ is $6u$. Since the cardinality of the set of pharmacophore $\mathcal{P}(m)$ of the molecule $m$ is $|m|^3$, the complexity of the direct computation of the pharmacophore kernel given in definition 21 is $(|m| \times |m'|)^3 \times 6u$. On the other hand, the computation given in Proposition 22 is a two step process :

- first is the initialization of the matrix $M$ : each of the $(|m| \times |m'|)^2$ entries is initialized by the product of a kernel $K_{Feat}$ with a kernel $K_{Dist}$, for a complexity of $(|m| \times |m'|)^2 \times 2u$
- second is the computation of the trace of $M^3$, which has a complexity of $(|m| \times |m'|)^3$

The global complexity of the matrix-based computation of the kernel is therefore $(|m| \times |m'|)^3 + (|m| \times |m'|)^2 \times 2u$, or equivalently $(|m| \times |m'|)^3 \times (1 + 2u/(|m| \times |m'|))$. In comparison with the direct approach, the matrix-based implementation proposed in Proposition 22 reduces the number of basis kernels $K_{Dist}$ and $K_{Feat}$ to be computed and is therefore more efficient. In any case, the complexity of the pharmacophore kernel computation is therefore $\mathcal{O}\left((|m| \times |m'|)^3\right)$. Even for relatively small molecules (of the order of 50 atoms), this complexity becomes in practice a serious issue when the size of the dataset increases to thousands or tens of thousands of molecules. However, we can note from the definition given in (4.8), that the lines of $M$ corresponding to pairs of points $(x, l) \in m$ and $(x', l') \in m'$ for which $K_{Feat}(l, l') = 0$ are filled with zeros. Based on this consideration, we observe that the cost of computing the kernel can be reduced by limiting the size of the matrix $M$, according to the following proposition.

**Proposition 23.** *If we let $M_2$ be the reduced version of a square matrix $M_1$, where the null lines and the corresponding columns are removed, then trace$(M_2^3)$ = trace$(M_1^3)$.*

*Proof.* Let $n_1$ (resp. $n_2$) be the size of $M_1$ (resp. $M_2$), and define $P$ (resp. $N$) as the subset of the set of indices $[1, n_1]$ that corresponds to the non-null (resp. null) lines of $M_1$. By definition, we have

$$\begin{aligned} \text{trace}(M_1^3) &= \sum_{i=1}^{n_1} M_1^3[i,i] \\ &= \sum_{i,j,k=1}^{n_1} M_1[i,j]M_1[j,k]M_1[k,i] \ . \end{aligned} \tag{4.9}$$

Moreover, if $i \in N$, then $M_1[i, j] = 0 \ \forall j \in [1, n_1]$. As a consequence, the term $M_1[i, j]M_1[j, k]M_1[k, i]$ in the summations over $i$, $j$, and $k$ in (4.9) is zero as soon as

at least one index $i$, $j$ or $k$ is in the set $N$. It follows that

$$\begin{aligned}\text{trace}(M_1^3) &= \sum_{i,j,k \in P} M_1[i,j]M_1[j,k]M_1[k,i] \\ &= \sum_{i,j,k=1}^{n_2} M_2[i,j]M_2[j,k]M_2[k,i] \\ &= \text{trace}(M_2^3) \ .\end{aligned}$$

■

Proposition 23 implies that the Cartesian product of $m$ and $m'$ involved in the matrix $M$ defined in (4.8) can be restricted to the pairs of points for which the label kernel $K_{Feat}$ is non-zero. In the case of the Dirac kernel (4.7) for discrete labels, this boils down to introducing a dimension in $M$ for any pair of atoms having the same label. This result can have important consequences in practice. Consider for example the case where the atoms of the molecules $m$ and $m'$ are uniformly distributed in $k$ classes of atom labels. In this case, the size of the matrix $M$ is equal to $k\,(|m|/k \times |m'|/k) = |m| \times |m'|/k$. The complexity of the kernel computation is therefore $\mathcal{O}\left((|m| \times |m'|/k)^3\right) = \mathcal{O}\left((1/k^3)(|m| \times |m'|)^3\right)$. It is therefore reduced by a factor $k^3$ in comparison with the original implementation. More generally this shows that important gains in memory and computation can be expected when the set of labels is increased. Section 4.5.3 discusses such a case in more details when the partial charges of atoms are included or not in the labels. Note finally that in a similar way, the kernel $K_{Dist}$ to compare distances can be set to a compactly supported kernel instead of the Gaussian RBF kernel (4.6). This has the effect of introducing sparsity in the matrix $M$, allowing the kernel computation to benefit from sparse matrix algorithms. This possibility was not further explored in this work.

## 4.3 Relation to graph kernels

In this section we show that the pharmacophore kernel can be seen as an extension of the walk-count graph kernels (Gärtner et al., 2003) to the 3D representation of molecules. The walk-count graph kernel is based on the representation of a molecule $m$ as a labeled graph $m = (\mathcal{V}, \mathcal{E})$, defined by a set of vertices $\mathcal{V}$, a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ connecting pairs of vertices, and a labeling function $l : \mathcal{V} \cup \mathcal{E} \to \mathcal{A}$, assigning a label $l(x)$ in an alphabet $\mathcal{A}$ to any vertex or edge $x$. In the case of molecules, the set of vertices $\mathcal{V}$ corresponds to the atoms of the molecule, and the edges of the graph are usually defined as the covalent bonds between the atoms of the molecules (Gärtner et al., 2003; Kashima et al., 2004; Mahé et al., 2005). In order to extend this 2D representation to a graph structure capturing 3D information, we propose to introduce an edge between any pair of vertices of the graph. Molecules are therefore seen as complete, atom-based graphs. If we now define a walk of length $n$ as a succession of $n+1$ connected

vertices, it is easy to see that there is a one-to-one correspondence between the set of pharmacophores $\mathcal{P}(m)$ of a molecule $m$, and its set of self-returning walks of length-three, defined formally for the molecule $m = (\mathcal{V}, \mathcal{E})$ as

$$\mathcal{W}_3^*(m) = \left\{(v_0, v_1, v_2, v_3) \in \mathcal{V}^4 : (v_i, v_{i+1}) \in \mathcal{E}, 0 \leq i \leq 2 \ \wedge \ v_3 = v_0\right\}.$$

We can therefore write the pharmacophore kernel (4.2) as a walk-based graph kernel:

$$K(m, m') = \sum_{p \in \mathcal{P}(m)} \sum_{p' \in \mathcal{P}(m')} K_{\mathcal{P}}(p, p') = \sum_{w \in \mathcal{W}_3^*(m)} \sum_{w' \in \mathcal{W}_3^*(m')} K_{Walk}(w, w') \ ,$$

where $K_{Walk}(w, w') = K_P(p, p')$ for the pair of walks $(w, w')$ corresponding to the pair of pharmacophores $(p, p')$. More precisely, consider a pair of pharmacophores $p = ((x_1, l_1), (x_2, l_2), (x_3, l_3))$ and $p' = ((x'_1, l'_1), (x'_2, l'_2), (x'_3, l'_3))$, and a corresponding pair of walks $w = (w_1, w_2, w_3, w_1)$ and $w' = (w'_1, w'_2, w'_3, w'_1)$. There is a direct equivalence between $K_{\mathcal{P}}$ and $K_{Walk}$ if we choose to label the vertices of the graphs by the atom labels involved in the pharmacophore characterization, and to label the edges by the Euclidean distance between the atoms they connect. Indeed, in this case we can write :

$$\begin{aligned} K_{\mathcal{P}}(p, p') &= \prod_{i=1}^{3} K_{Feat}(l_i, l'_i) \, K_{Dist}\left(\| x_i - x_{i+1} \|, \| x'_i - x'_{i+1} \|\right) \\ &= \prod_{i=1}^{3} K_{Feat}(l(w_i), l(w'_i)) \, K_{Dist}\left(l((w_i, w_{i+1})), l\left((w'_i, w'_{i+1})\right)\right) \\ &= K_{Walk}(w, w') \end{aligned}$$

A striking point of this kernel between walks is that it can be factorized along the edges of the walks:

$$\begin{aligned} K_{Walk}(w, w') &= \prod_{i=1}^{3} K_{Feat}(l(w_i), l(w'_i)) \, K_{Dist}\left(l((w_i, w_{i+1})), l\left((w'_i, w'_{i+1})\right)\right) \\ &= \prod_{i=1}^{3} K_{Step}\left((w_i, w_{i+1}), (w'_i, w'_{i+1})\right) \end{aligned} \tag{4.10}$$

The pharmacophore kernel therefore formulates as a walk-based graph kernel, with a walk kernel factorizing along the edges of the walks. It follows from Kashima et al. (2004) that it can be computed by the formalism based on product-graphs and powers of the adjacency matrix proposed in Gärtner et al. (2003), if the adjacency matrix of the product-graph is weighted by the walk-step kernels $K_{Step}$ (4.10). Consequently, the matrix $M$ defined in (4.8) and upon which is based the kernel computation of Proposition 22, can be seen as a weighted adjacency matrix of a product-graph defined on complete, atom-based, molecular factor graphs.

Note moreover that in its way to characterize the molecular structure, the pharmacophore kernel bears some similarity with cyclic patterns kernels (Horváth et al., 2004) where a molecule is represented by graph cycles, even though general cycles instead of cycles of size three are considered in this latter approach, and the graph corresponds to the 2D structure of the molecule.

## 4.4 Fast approximation

As an alternative to the costly computation presented in Section 4.2, we introduce in this section a fast approximation to the pharmacophore kernel based on a discretization of the pharmacophore space. Our definition of pharmacophores is based on the atoms 3D coordinates, but they can equivalently be characterized by the pairwise distances between atoms. In order to define discrete pharmacophores, we restrict ourselves to discrete sets of atom labels (e.g., the set of atom types), and we discretize uniformly the range of distances between atoms into a predefined number of bins. For example, if the inter atomic distances lie in the 0-20 angstroms () range, and we consider 10 bins to discretize the distances, the bins will correspond to the intervals 0-2, 2-4,...,18-20. Each distance is then mapped to the index of the bin it falls in, and a discrete pharmacophore is defined by a triplet of atom labels together with a triplet of bin indices. More formally, if the distance range is discretized into $p$ bins, the set of discrete pharmacophores is a finite set defined as $\mathcal{T}_3 = \mathcal{L}^3 \times [1, p]^3$, where $\mathcal{L}$ is the set of atom labels.

Consider the mapping $\phi^{3pt}$ from the set of molecules to the set of discrete pharmacophores $\mathcal{T}_3$, defined for the molecule $m$ as $\phi^{3pt}(m) = (\phi_t(m))_{t \in \mathcal{T}_3}$, where $\phi_t(m)$ is the number of times the pharmacophore $t$ is found in the molecule $m$. This mapping leads to the following kernel definition.

**Definition 24 (Three-points spectrum kernel).** *For a pair of molecules $m$ and $m'$, we define the* three-points spectrum kernel $K^{3pt}_{Spec}$ *as*

$$K^{3pt}_{Spec}(m, m') = \langle \phi^{3pt}(m), \phi^{3pt}(m') \rangle = \sum_{t \in \mathcal{T}_3} \phi_t(m) \phi_t(m') \ . \tag{4.11}$$

Note that if we define the mapping $d : \mathcal{P} \mapsto \mathcal{T}_3$, such that $d(p)$ is the discretized version of the pharmacophore $p \in \mathcal{P}$, we can explicitly write the three-points spectrum kernel as a particular pharmacophore kernel (4.2):

$$K^{3pt}_{Spec}(m, m') = \sum_{p \in \mathcal{P}(m)} \sum_{p' \in \mathcal{P}(m')} \mathbf{1} \left( d(p) = d(p') \right) \ . \tag{4.12}$$

This equation shows that this is a crude pharmacophore kernel, based on a kernel for pharmacophores that simply checks if two given pharmacophores have identical discretized versions or not.

In addition, we consider a "two-points pharmacophore" version of the kernel (4.11), based on pairs, instead of triplets, of atoms (Swamidass et al., 2005). Letting $\mathcal{T}_2$ be the set of all possible two-points pharmacophores, that is, pairs of atom types together with the bin index of the edge connecting them, and $\phi^{2pt}(m) = (\phi_t(m))_{t\in\mathcal{T}_2}$ be the mapping of the molecule $m$ to $\mathcal{T}_2$, corresponding to $\phi^{3pt}(m)$ , we define the following *two-points spectrum* kernel.

**Definition 25 (Two-points spectrum kernel).** *For a pair of molecules $m$ and $m'$, we define the* two-points spectrum kernel $K^{2pt}_{Spec}$ *as :*

$$K^{2pt}_{Spec}(m,m') = \langle \phi^{2pt}(m), \phi^{2pt}(m') \rangle = \sum_{t\in\mathcal{T}_2} \phi_t(m)\phi_t(m') \ . \tag{4.13}$$

This kernel shows strong similarities with recently introduced kernels for 3D structures of molecules (Swamidass et al., 2005), and is introduced as a baseline to validate the three-points pharmacophore characterization of molecules.

The kernels (4.11) and (4.13) are directly expressed as dot-products, and are consequently positive definite, which justifies their use with SVM. Moreover, these kernels can be computed efficiently using an algorithm derived from that used in the implementation of spectrum string kernels (Leslie et al., 2002). We describe this algorithm for the three-points version of the kernel, its extension to the two-points kernel being straightforward. Following the notation of Section 4.3, we represent molecules by complete, atom-based labeled graphs, with the difference that the set of atom labels $\mathcal{L}$ defining the vertices labels is considered to be discrete (e.g., the atom types), and the edges are now labeled by the bin index of the corresponding inter-atomic distance. We consider the problem of computing the Gram matrix $K$ associated to such a set of molecular graphs $\left\{G_i = (\mathcal{V}_{G_i}, \mathcal{E}_{G_i})\right\}_{i=1,\ldots,n}$ for the kernel (4.11). The alphabet $\mathcal{A}$, involved in the graph labeling function $l$ of section 4.3, is defined as $\mathcal{A} = \mathcal{A}_V \cup \mathcal{A}_E$, where $\mathcal{A}_V$ is the set of vertex labels, corresponding to the set of atom labels $\mathcal{L}$, and $\mathcal{A}_E$ is the set of edges labels, corresponding to the set of distance bins indices.

The algorithm is based on the manipulation of sets of walk pointers within each graph, according to a tree transversal process. If we let $n$ and $p$ be the cardinalities of $\mathcal{A}_V$ and $\mathcal{A}_E$ respectively, we define a rooted tree of depth five structuring the space of pharmacophores $\mathcal{T}_3$ as follows :

- the root node[1] has $n$ sons, corresponding to the $n$ possible vertex labels
- the nodes of depth two and three have $n \times p$ sons, corresponding to the $n \times p$ possible pairs of edge and vertex labels
- the nodes of depth four have $p$ sons, corresponding to the $p$ possible edges labels, a leaf node being implicitly associated the vertex label of its depth-one ancestor.

[1]Recall from Chapter 3 that, by definition, the depth of the root node is 1.

A path from the root to a leaf node therefore corresponds to a triplet of distinct vertex labels, together with a triplet of distinct edge labels. There is therefore a one-to-one correspondence between the leaf nodes and the pharmacophores of $\mathcal{T}_3$. The principle of the algorithm is to recursively transverse this tree until each leaf node (i.e., each potential pharmacophore) is visited. During this process, a set of walk pointers is maintained within each molecule. The pointers are recursively updated such that the pointed walks correspond to the pharmacophores under construction in the tree-transversal process. When reaching a leaf node, the pointed walks correspond to the occurrences of a particular pharmacophore $t$ in the molecules. The mapping $\phi_t(G_i)$ can therefore be computed for the molecular graphs $\{G_i\}_{i=1,\ldots,n}$, and the kernel matrix $K$ can be updated by adding the products $\phi_t(G_i)\phi_t(G_j)$ to its $(i,j)$ entries.

A pseudo code of the algorithm is given in Algorithms 1, 2, 3 and 4. Algorithm 1 is the main program in charge of the tree-transversal process, and Algorithms 2, 3 and 4 are subroutines, introduced to initialize the walks pointers, extend the pointed walks, and update the Gram matrix respectively. This pseudo-code relies on the abstract types *Pointer* and *Label*, to represent the walk pointers involved in the algorithm, and the generic vertices and edges labels, belonging to $\mathcal{A}_V$ and $\mathcal{A}_E$ respectively. Formally, a *Pointer* object consists of two graph vertices: a *start* and *current* vertex, representing the first and the current vertices of the pointed walk under extension. To maintain walks pointers within each molecule, we introduce a matrix of pointers *walkPointers = Pointer[][]* : this matrix is initially empty, and during the walk extension process, *walkPointers[i][j]* corresponds to the $j$-th pointer of the molecular graph $G_i$. The stopping criterion of the recursion is controlled by an integer variable *depth* corresponding to the depth in the tree during the transversal process. It is initialized to one and incremented at each recursive call. When *depth* is four, a node of depth four is reached in the tree, which corresponds to pointers on walks of length two in the graphs. In the subsequent recursive step, *depth* is five, and the pointers are updated to ensure that the extended walks correspond to self-returning ones. A leaf node is then reached and the recursion terminates, leading to an update of the Gram matrix. Note however that the recursion is aborted whenever the set of walk pointers becomes empty for all graphs, since we only need to reach the leaf nodes corresponding to the pharmacophores truly present in the set of graphs.

Computing the Gram matrix $K$ simply requires a call to the *COMPUTE* function of Algorithm 1 with the following arguments: *walkPointers*, the empty Pointer matrix, *depth* initialized to one, and $K$, the $n \times n$ Gram matrix filled with zeros. The cost of this algorithm depends on the number of leaf nodes visited, and is therefore bounded by the total number of leaves of the tree, that is $(np)^3$ if the number of distinct vertex labels is $n$ and the number of distance bins is $p$. However, the maximum number of distinct pharmacophores that can be found in the molecule $m$ is $|m|^3$, and we do not need to exhaustively transverse the tree. This means that to compute the kernel between the molecules $m$ and $m'$, at most $\min(|m|^3, |m'|^3)$ leaves, corresponding to the common pharmacophores of $m$ and $m'$, need to be visited. The complexity of the algorithm is therefore $\mathcal{O}\left(\min\left((np)^3, \min\left(|m|^3, |m'|^3\right)\right)\right)$. For small molecules, the cost

of the kernel will therefore depend on their number of atoms, while it will depend on the size of the discrete pharmacophore space for large molecules.

Note finally that although we omit the details, the previous algorithm and complexity analysis hold for the two-points versions of the kernels : the tree involved in the recursive transversal process is smaller (a tree of depth three with $n^2p$ leaf nodes), and the complexity is reduced to $\mathcal{O}\left(\min\left(n^2p, \min(|m|^2, |m'|^2)\right)\right)$.

**Algorithm 1** main program

```
COMPUTE(Pointer[][] walkPointers,Integer depth, Float[][] K)
  depth = depth + 1
  if depth = 2 then
    for label ∈ A_V do
      walkPointers = initPointers(label)
      compute(walkPointers, depth, K)
    end for
  else
    for label_1 ∈ A_V do
      for label_2 ∈ A_E do
        walkPointers = extendPointers(walkPointers, depth, label_1, label_2)
        if walkPointers ≠ [][] then
          if depth = 5 then
            updateGram(walkPointers, K)
          else
            compute(walkPointers, depth, K)
          end if
        end if
      end for
    end for
  end if
```

## 4.5 Experiments

We now turn to the experimental section. The problem considered here consists in building predictive models to distinguish *active* from *inactive* molecules on several protein targets. This problem is naturally formulated as a supervised binary classification problem that can be solved by SVM.

### 4.5.1 Datasets

We tested the pharmacophore kernel on several datasets used in a recent SAR study (Sutherland et al., 2003). More precisely, we considered the following four publicly

**Algorithm 2** Sub-routine 1 : initialize walks pointers

INITPOINTERS(Label *label*)

```
walkPointers = Pointer[][]
for i = 1, ..., n do
  for v ∈ V_{G_i} do
    if l(v) = label then
      walkPointers[i].addPointer(start = v, current = v)
    end if
  end for
end for
return walkPointers
```

**Algorithm 3** Sub-routine 2 : extend walks pointers

EXTENDPOINTERS(Pointer[][] $walkPointers_{in}$, Integer *depth*, Label $label_1$, Label $label_2$)

```
walkPointers_out = Pointer[][]
for i = 1, ..., n do
  for ptr ∈ walkPointers_in[i] do
    for (ptr.current, v) ∈ E_{G_i} do
      if l(v) = label_1 ∧ l((ptr.current, v)) = label_2 then
        if depth ≠ 5 ∨ v = ptr.start then
          walkPointers_out[i].addPointer(start = ptr.start, current = v)
        end if
      end if
    end for
  end for
end for
return walkPointers_out
```

**Algorithm 4** Sub-routine 3 : update Gram matrix

UPDATEGRAM(Pointer[][] *walkPointers*, Float [][] *K*)

```
for i = 1, ..., n do
  for j = 1, ..., n do
    if walkPointers[i] ≠ [] ∧ walkPointers[j] ≠ [] then
      K[i][j] = K[i][j] + walkPointers[i].size() × walkPointers[j].size()
      if i ≠ j then
        K[j][i] = K[j][i] + walkPointers[i].size() × walkPointers[j].size()
      end if
    end if
  end for
end for
```

| | TRAIN | | TEST | |
|---|---|---|---|---|
| | Pos | Neg | Pos | Neg |
| BZR | 94 | 87 | 63 | 62 |
| COX | 87 | 91 | 61 | 64 |
| DHFR | 84 | 149 | 42 | 118 |
| ER | 110 | 156 | 70 | 110 |

Table 4.1: Basic information about the datasets considered.

available datasets [2]:

- the *BZR* dataset, a set of 405 ligands for the benzodiazepine receptor,
- the *COX* dataset, a set of 467 cyclooxygenase-2 inhibitors,
- the *DHFR* dataset, a set of 756 inhibitors of dihydrofolate reductase,
- the *ER* dataset, a set of 1009 estrogen receptor ligands.

These datasets contain the 3D structures of the molecules, together with a quantitative measure of their ability to inhibit a biological mechanism. The reference paper (Sutherland et al., 2003) presents a data preparation scheme sought to mimic a real virtual screening application: datasets were first filtered to prevent structural redundancy in the compounds considered, and were further split in training and test sets such that the compounds used for testing are as structurally different as possible to those used for training. In order to have a reference result to compare to, we kept this particular data preparation scheme. Table 4.1 gathers basic informations about the datasets involved in the study.

### 4.5.2 Experimental setup

We investigated in this study a simple labeling scheme to describe each atom (hydrogen atoms were systematically removed), and therefore the potential pharmacophores: the label of an atom is composed of its type (e.g., $C$, $O$, $N$...) and the sign of its partial charge ($+, -$ or 0). Hence the set of labels can be expanded as $\mathcal{L} = \{C^+, C^0, C^-, O^+, O^0, O^-, \ldots\}$. The partial charges account for the contribution of each atom to the total charge of the molecule, and were computed with the QuacPAC software developed by OpenEye [3]. It is important to note that, contrary to the physicochemical properties of atoms, partial charges depend on the molecule and describe the spatial distribution of charges. Although the partial charges take continuous values, we simply kept their signs for the labeling as basic indicators of charges in the

[2] Available as supporting information of the original study at `http://pubs.acs.org/journals/jcisd8/`

[3] `http://www.eyesopen.com/products/applications/quacpac.html`

description of pharmacophores. We call *categorical kernel* the kernel resulting from this labeling, where the kernel between labels $K_{Feat}$ is the Dirac kernel (4.7) and the kernel between distances $K_{Dist}$ is the Gaussian RBF kernel (4.6).

Alternatively, we tested several variants of this basic categorical kernel. On the one hand, we tested the effect of the partial charges by removing them from the labels, and keeping the same Dirac and Gaussian RBF kernels for the labels and distances, respectively. In this case the label of an atom reduces to its type. On the other hand, we tested the fast approximation and its two-points counterpart mentioned in Section 4.4 with our original labeling scheme, that is, atoms labeled by their types and the sign of their partial charges.

In addition, we tested the state-of-the-art Tanimoto kernel based on the 2D structure of molecules (Ralaivola et al., 2005) to evaluate the potential gain obtained by including 3D information. This kernel is defined as the Tanimoto coefficient between fingerprints indicating the presence or absence of all possible molecular fragments of length up to 8 in the 2D structure of the molecule, where a fragment refers to a sequence of atoms connected by covalent bonds. We note that this fingerprint is similar to classical 2D-fingerprints such as the Daylight representation[4], with the difference that our implementation does not require to fold the fingerprint into a small-size vector.

The different kernels were implemented in C++ within the open-source ChemCpp toolbox[5], and the SVM experiment was conducted with the open-source Python machine learning package PyML[6]. For each experiment, all parameters of the kernel and the SVM were optimized over a grid of possible choices on the training set only, to maximize the mean AUC (see Section 1.3.1) over an internal 10-fold cross-validation. The results on the test set correspond to the performance of the SVM with the selected parameters only. The optimized parameters include the width $\sigma \in \{0.1, 1, 10\}$ (in angstroms) of the Gaussian RBF kernel used to compare distances, the soft-margin parameter of the SVM over the grid $\{0.1, 0.5, 1, 1.5, ..., 20\}$, and the number of bins used to discretize the distances for the fast approximations over the grid $\{4, 6, 8, \ldots, 30\}$.

### 4.5.3 Results

Table 4.3 shows the results of classification for the different kernel variants. Each line corresponds to a kernel, and reports several statistics : the accuracy (fraction of correctly classified compounds), sensitivity (fraction of positive compounds that were correctly classified), specificity (fraction of negative compounds that were correctly classified), and AUC. The first line corresponds to the basic categorical kernel. The following three lines show the results of the variants of the categorical kernel: the reduction of the atom labels to their types (i.e., categorical kernel without partial charges), and the fast approximation of the kernel (i.e., three-points spectrum kernel), together with its two-points counterpart. Finally, we added the performance obtained

[4] `http://www.daylight.com/dayhtml/doc/theory/theory.toc.html`

[5] Available at `http://chemcpp.sourceforge.net`

[6] Available at `http://pyml.sourceforge.net`

by the state-of-the-art 2D Tanimoto kernel, based on the 2D structure of the molecules, and the best results reported in the reference publication (Sutherland et al., 2003). This latter method, labeled *"Sutherland"* in table 4.3, is based on descriptors inherited from the 2D structure and the atomic composition of the molecules, that are selected using a genetic algorithm.

The results of parameters optimization on the training set often led to similar choices for different kernels. For example, the width of the Gaussian RBF kernel to compare distances was usually selected at 0.1 angstrom, which corresponds to a very strong constraint on the pharmacophore matching. Finally, the number of bins selected by the fast approximations to discretize the distances was usually between 20 and 30 bins.

We can first observe from table 4.3 that removing the partial charges from atom labels decreases the accuracy by 2 to 4%, corresponding to a relative variation of 3 to 5%, on all datasets except COX. This superiority in accuracy of the categorical kernel is significant at a p-value $p = 0.125$, according to the one-sided Wilcoxon signed-rank test for paired data (Demšar, 2006) based on the accuracy statistic, which suggests that the partial charge information is important for the definition of pharmacophores.

Moreover, the fast pharmacophore kernel obtained by applying a Dirac kernel to check when pairs of candidate pharmacophores fall in the same bin of the discretized space (three-points spectrum kernel) systematically degrades the accuracy by 1 to 5%, corresponding to a relative variation of 1 to 6%, over all four datasets compared to the categorical kernel. This is significant at a p-value $p = 0.062$, and suggests that the gain in computation time obtained by discretizing the space and computing a 3D-fingerprint-like representation of molecules has a cost in terms of accuracy of the final model. A particular limitation of the fingerprint-based method is that two pharmacophores could remain unmatched if they fall into two different bins, although they might be very similar but close to the bins boundaries. In the case of the pharmacophore kernel, such pairs of similar pharmacophores would always be matched.

We observe finally that except for the COX dataset, the discrete kernel based on two-points pharmacophores lead to worse accuracy results than its three-points counterpart. This tends to highlight the benefits of the three-points pharmacophore characterization of the molecular structure, but this is only significant at a p-value $p = 0.312$.

For each dataset, the results obtained with the 2D-Tanimoto kernel are significantly worse than those of the categorical kernel, with a decrease ranging from 3 to 7%, corresponding to a relative variation of 3 to 10%, on the different datasets. This is significant at a p-value $p = 0.062$ and confirms the relevance of 3D information for drug activity prediction, that motivated this work. Finally we note that on all but the COX dataset, the categorical kernel outperforms the best results of Sutherland et al. (2003). This tends to confirm the competitiveness of our method compared to state-of-the-art methods, but these latter results are only significant at a p-value $p = 0.312$.

Regarding the computational complexity of the different methods, Table 4.2 shows the time required to compute the kernel matrices on the BZR training set for different

| | Exact | Discrete |
|---|---|---|
| With charges | 20' | 6' |
| Without charges | 249' | 7' |

Table 4.2: Computation times in minutes needed to compute the different kernel matrices on the BZR training set. The first column refers to the computation of the exact kernel (4.3), and the second one to the approximate kernel (4.11).

kernels, on a desktop computer, equipped with a Pentium 4 - 3.6 GHz processor, and 1 GB RAM. In the discrete version, the distance range was split into 24 bins, and as expected, the kernels based on the discretization of the pharmacophore space are faster than their counterparts by a factor of 4 to 35, depending on the type of labels used (with or without the partial charge information). In the exact kernel computation, the effect of removing the partial charges from the labels is to induce more matches between atoms and therefore, as discussed in Section 4.2, to drastically slow the computation by a factor of 12, consistent with the theoretical estimate that dividing the size of the label classes by $k$ increases the speed by a factor $k^3$.

## 4.6 Discussion and conclusion

This paper presents an attempt to extend the application of recent machine learning algorithms for classification to the manipulation of 3D structures of molecules. This attempt is mainly motivated by applications in drug activity prediction, for which 3D pharmacophores are known to play important roles. Although previous attempts to define kernels for 3D structures (similar in fact to the two-points spectrum kernel we tested) led to mixed results (Swamidass et al., 2005), we obtained performance competitive with state-of-the-art algorithms for the categorical kernel based on the comparison of pharmacophores contained in the two molecules to be compared. This kernel is not an inner product between fingerprints, and therefore fully exploits the mathematical trick that allows SVM to manipulate measures of similarities rather than explicit vector representations of molecules, as opposed to other methods such as neural networks. We even observed that for the closest fingerprint-based approximation obtained by discretizing the space of possible pharmacophores (three-points spectrum kernel), the performance significantly decreases. This highlights the benefits that can be gained from the use of kernels, which provide a satisfactory answer to the common issue of choosing a "good" discretization of the pharmacophore space to make fingerprints: once discretized, pharmacophores falling on different sides of bins edges do not match although they might be very close. We notice that approaches based on fuzzy fingerprints (Horvath and Jeandenans, 2003), for example, aim at correcting this effect by matching pharmacophores based on different distance bins.

Among the possible extensions to our work, a promising direction that is likely

to be relevant for many real-world applications is to take into accounts different conformers of each molecule. Indeed, it is well-known that the biological activity to be predicted is often due to one out of several conformers for a given molecule, which suggests to represent a molecule not as a single 3D structure but as a set of structures. This problem, known as multi-instance learning, has been drawing considerable interest in the machine learning community since its initial formulation (Dietterich et al., 1997). The SVM and kernel approaches lend themselves particularly well to this extension, thanks to the possibility to define kernels between sets of structures from a kernel between structures (Gärtner et al., 2002), and extensions of the SVM algorithm (Andrews et al., 2002). A second direction would be to test and validate different definitions and labeling for the vertices of the pharmacophores. We limited ourselves to the simplest possible three-points pharmacophores based on single atoms annotated by their types and partial charges. The method could be improved by testing other schemes known to be relevant features as basic components of pharmacophores. It is for example possible to consider groups of atoms forming functional units instead of single atoms to form pharmacophores. Alternatively, the atom labels considered in this work may be enriched with the introduction of various physicochemical properties known to account for the steric and electrostatic behavior of atoms. As a first step in this direction, we investigated a labeling scheme based on a set of four physicochemical properties (namely the atomic Van der Waals and covalent radii, electronegativity and first ionization energy) but the corresponding results were not convincing: they were globally similar to those obtained with atom types labels without partial charges. This is actually not really surprising because these properties are deduced from the atom types, and therefore bear little additional information, contrary to the partial charges which depend on the molecular conformation. A third possible extension is to generalize this work to pharmacophores with more points, e.g., 4 or 5. Although several results will not remain valid in this case, such as the expression of the kernel as the trace of a matrix, this could lead to more accurate models in cases where the binding mechanism is well characterized by such pharmacophores. Finally, we note that several approaches were recently proposed to derive a measure of similarity between structured objects from the similarity of their substructures(Cuturi and Vert, 2005; Wolf and Shashua, 2003; Jebara et al., 2004). These approaches could generalize the present work to alternative measures of similarity between 3D structures based on pharmacophore similarity.

| | BZR | | | | COX | | | | DHFR | | | | ER | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Sens. | Spec. | AUC | Acc. | Sens. | Spec. | AUC | Acc. | Sens. | Spec. | AUC | Acc. | Sens. | Spec. | AUC |
| Categorical | **76.4** | 74.0 | 78.9 | 82.1 | 69.8 | 69.8 | 69.8 | 75.1 | **81.9** | 63.3 | 88.8 | 84.8 | **79.8** | 72.0 | 84.7 | 86.8 |
| Categorical, no partial charges | 74.3 | 73.6 | 75.0 | 81.5 | 70.0 | 68.5 | 70.9 | 74.6 | 78.1 | 65.2 | 82.7 | 82.2 | 77.6 | 71.7 | 81.4 | 87.2 |
| Three-points spectrum | 75.4 | 74.4 | 76.3 | 81.3 | 67.0 | 64.4 | 69.5 | 75.9 | 76.9 | 70.9 | 79.0 | 81.9 | 78.6 | 78.3 | 78.8 | 87.4 |
| Two-points spectrum | 71.4 | 61.3 | 81.6 | 80.3 | 68.9 | 70.2 | 67.7 | 74.7 | 67.7 | 67.4 | 67.9 | 72.3 | 78.7 | 75.9 | 80.4 | 84.5 |
| 2D-Tanimoto | 71.2 | 71.9 | 70.5 | 80.8 | 63.0 | 67.5 | 58.6 | 69.8 | 76.9 | 73.8 | 78.0 | 83.0 | 77.1 | 69.3 | 82.1 | 83.6 |
| Sutherland et al. (2003) | 75.2 | 70.0 | 81.0 | XXX | **73.6** | 75.0 | 72.0 | XXX | 71.9 | 74.0 | 71.0 | XXX | 78.9 | 77.0 | 80.0 | XXX |

Table 4.3: Classification of the test sets, after model selection on the training set.

# Conclusion

The achievements of this thesis suggest, in our opinion, that the kernel approach looks promising to model Structure-Activity Relationships. Moreover, we believe that kernel functions between molecular structures may offer, to some extent, a unified approach to SAR and virtual screening, for two reasons. First, because they circumvent the need for selecting and extracting molecular descriptors, such kernels can straightforwardly be used to model different biological properties. Second, because of the modularity of kernel methods, these kernels can be used in conjunction with several existing kernel algorithms, in order to solve various tasks such as, for instance, regression, clustering and similarity analysis. However, this thesis has to be seen as a preliminary study, that should be validated by other experiments, involving for instance different learning problems and other databases. In particular, because large databases are commonplace in chemoinformatics, the kernel approach would benefit from a large-scale validation involving larger databases than those that have been considered in this thesis. Concerning the practical use of our approach for screening of large datasets, we observed that the approach based on kernel methods can be computationally demanding even for relatively small datasets. In practice, the time to train SVM can be reduced because not all entries of the matrix are required. 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.

We see several possible extensions to our work. On the practical side, the fact that the models could benefit from simple graph enrichments based on Morgan indices and partial charges, suggests that the introduction of a more thorough chemical knowledge may 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 part of the same functional groups in the 2D representation (Gillet et al., 2003), and in the 3D case by considering pharmacophoric features instead of isolated atoms (Pickett et al., 1996). Applying such transformations in a pre-processing step is likely to improve the characterization of the molecular structures in the kernels, and at the same time reduce their computation cost, that depends on the size of the graphs to be compared. On the methodological side, different ways are probably worth exploring. A first extension would be to adopt a global representation of molecules and integrate information derived from their 1D, 2D and 3D structures. A possible

approach toward this goal would be to consider a single kernel defined as a linear combination of kernels for 2D and structures, together with simple kernels based on global physicochemical properties. Several methods have been proposed in order to optimize such a kernel combination based, for instance, on semi-definite programming (Lanckriet et al., 2004). Another important extension to the pharmacophore kernel would be to consider a multi-conformers representations of 3D molecular structures. Indeed, the restriction to a single conformation is clearly an unrealistic assumption, and we believe that this extension is likely to make sense for real-world applications. On the methodological side, the introduction of multi-conformers would cast the learning problem into the framework of multiple-instance learning, and, as suggested at the end of Chapter 4, the SVM and kernel approaches lend themselves well to this extension. In particular, the pharmacophore kernel could be extended to a kernel between sets of structures using the framework of multi-instance kernels (Gärtner et al., 2002), or alternatively, it could used directly in conjunction with a multi-instance extension of the SVM algorithm (Andrews et al., 2002). We note also that real-world applications based on a sequential screening process, could benefit from advances in the transductive and active approaches to learning, that were initially introduced for virtual screening applications by Weston et al. (2003) and Warmuth et al. (2003). These two approaches might in particular be used in conjunction in order to prioritize the screening of pre-defined virtual libraries. Finally, we believe that this work can find many other applications in various domains where data naturally come as graphs, such as bioinformatics, natural language processing or digital image analysis for instance.

# Bibliography

M. A-Razzak and R. C. Glen. Applications of rule-induction in the derivation of quantitative structure-activity relationships. *J Comput Aided Mol Des*, 6(4):349–383, Aug 1992.

E. Abrahamian, P. C. Fox, L. Naerum, I. T. Christensen, H. Thøgersen, and R. D. Clark. Efficient generation, storage, and manipulation of fully flexible pharmacophore multiplets and their use in 3-D similarity searching. *J Chem Inf Comput Sci*, 43(2):458–468, 2003. URL `http://dx.doi.org/10.1021/ci025595r`.

T. A. Andrea and H. Kalayeh. Applications of neural networks in quantitative structure-activity relationships of dihydrofolate reductase inhibitors. *J Med Chem*, 34(9):2824–2836, Sep 1991.

S. Andrews, T. Hofmann, and I. Tsochantaridis. Multiple Instance Learning with Generalized Support Vector Machines. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 943–944. American Association for Artificial Intelligence, 2002.

T. Aoyama, Y. Suzuki, and H. Ichikawa. Neural networks applied to quantitative structure-activity relationship analysis. *J Med Chem*, 33(9):2583–2590, Sep 1990.

M. Arakawa, K. Hasegawa, and K. Funatsu. Application of the novel molecular alignment method using the Hopfield Neural Network to 3D-QSAR. *J Chem Inf Comput Sci*, 43(5):1396–1402, 2003. URL `http://dx.doi.org/10.1021/ci030005q`.

J. W. Armstrong. A review of high-throughput screening approaches for drug discovery. Application note, 1999.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950.

J. Bajorath. Selected concepts and investigations in compound classification, molecular descriptor analysis, and virtual screening. *J Chem Inf Comput Sci*, 41(2):233–245, 2001.

J. Bajorath. Integration of virtual and high-throughput screening. *Nat Rev Drug Discov*, 1(11):882–894, Nov 2002. URL `http://dx.doi.org/10.1038/nrd941`.

S.C. Basak, V.R. Magnuson, G.J. Niemi, and R.R. Regal. Determining Structural Similarity of Chemicals Using Graph Theoretic Indices. *Discrete Appl. Math.*, 19: 17–44, 1988.

H. Bauknecht, A. Zell, H. Bayer, P. Levi, M. Wagener, J. Sadowski, and J. Gasteiger. Locating biologically active compounds in medium-sized heterogeneous datasets by topological autocorrelation vectors: dopamine and benzodiazepine agonists. *J Chem Inf Comput Sci*, 36(6):1205–1213, 1996.

K. H. Bleicher, H.-J. Böhm, K. Müller, and A. I. Alanine. Hit and lead generation: beyond high-throughput screening. *Nat Rev Drug Discov*, 2(5):369–378, May 2003. URL `http://dx.doi.org/10.1038/nrd1086`.

H.-J. Böhm, G. Schneider, R. Mannhold, H. Kubinyi, and G. Folkers. *Protein-ligand interactions*. Wiley, 2003.

A. Boobis, U. Gundert-Remy, P. Kremers, P. Macheras, and O. Pelkonen. In silico prediction of ADME and pharmacokinetics. Report of an expert meeting organised by COST B15. *Eur J Pharm Sci*, 17(4-5):183–193, Dec 2002.

K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. *Int. Conf on Data Mining*, 0:74–81, 2005.

K.M. Borgwardt, C.S. Ong, S. Schönauer, S.V.N. Vishwanathan, A.J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl. 1): i47–i56, Jun 2005. URL `http://dx.doi.org/10.1093/bioinformatics/bti1007`.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992. URL `http://www.clopinet.com/isabelle/Papers/colt92.ps.Z`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

R. D. Brown and Y. C. Martin. The information content of 2D and 3D structural descriptors relevant to ligand-receptor binding. *J Chem Inf Comput Sci*, 37:1–9, 1997.

Robert D. Brown and Yvonne C. Martin. Use of Structure-Activity Data To Compare Structure-Based Clustering Methods and Descriptors for Use in Compound Selection. *J Chem Inf Comput Sci*, 36:572–584, 1996.

P. Bultinck, T. Kuppens, X. Gironès, and R. Carbó-Dorca. Quantum similarity superposition algorithm (QSSA): a consistent scheme for molecular alignment and molecular similarity based on quantum chemistry. *J Chem Inf Comput Sci*, 43(4): 1143–1150, 2003. URL `http://dx.doi.org/10.1021/ci0340153`.

H. Bunke and K. Shearer. A Graph Distance Metric based on the Maximal Common Subgraph. *Pattern Recognition Letters*, 19:255–259, 1998.

R. Burbidge, M. Trotter, B. Buxton, and S. Holden. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comput. Chem.*, 26(1):4–15, December 2001. URL `http://stats.ma.ic.ac.uk/~rdb/pubs/candc-aisb00-rbmt-final.pdf`.

C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998. URL `http://www.kernel-machines.org/papers/Burges98.ps.gz`.

D. Butina, M. D. Segall, and K. Frankcombe. Predicting ADME properties in silico: methods and models. *Drug Discov Today*, 7(11 Suppl):S83–S88, Jun 2002.

E. Byvatov, U. Fechner, J. Sadowski, and G. Schneider. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *J Chem Inf Comput Sci*, 43(6):1882–9, 2003. URL `http://dx.doi.org/10.1021/ci0341161`.

Evgeny Byvatov and Gisbert Schneider. SVM-based feature selection for characterization of focused compound collections. *J Chem Inf Comput Sci*, 44(3):993–9, 2004. URL `http://dx.doi.org/10.1021/ci0342876`.

R. Carbó, L. Leyda, and M. Arnau. How Similar is a Molecule to Another - an Electron-Density Measure of Similarity Between 2 Molecular Structures. *Int. J. Qantum Chem.*, 17:1185–1189, 1980.

R.E. Carhart, D.H. Smith, and R. Venkataraghavan. Atom Pairs as Molecular Features in Structure-Activity Studies: Definitions and Applications. *J Chem Inf Comput Sci*, 25:64–73, 1985.

S. K. Chanda and J. S. Caldwell. Fulfilling the promise: drug discovery in the post-genomic era. *Drug Discov Today*, 8(4):168–174, Feb 2003.

X. Chen, A. Russinko III, and S. S. Young. Recursive Partitioning Analysis of a Large Structure-Activity Data Set Using Three-Dimensional Descriptors. *J Chem Inf Comput Sci*, 38:1054–1062, 1998.

M. Collins and N. Duffy. Convolution Kernels for Natural Language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Adv. Neural. Inform. Process Syst.*, volume 14, pages 625–632. MIT Press, 2001.

M. Cuturi and J.-P. Vert. Semigroup kernels on finite sets. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Adv. Neural Inform. Process. Syst.*, volume 17, pages 329–336. MIT Press, Cambridge, MA, 2005.

A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Schusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

C. Debouck and P. N. Goodfellow. DNA microarrays in drug discovery and development. *Nat Genet*, 21(1 Suppl):48–50, Jan 1999. URL `http://dx.doi.org/10.1038/4475`.

J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.*, 7:1–30, 2006. URL `http://jmlr.csail.mit.edu/papers/v7/demsar06a.html`.

M. Deshpande and G. Karypis. Automated Approaches for Classifying Structures. In *Proceedings of the 2nd Workshop on Data Mining in Bioinformatics (BIOKDD '02), 2002*, 2002.

M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent Substructure-Based Approaches for Classifying Chemical Compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1036–1050, August 2005.

J. Devillers. *Neural Networks in QSAR and Drug Design.* Academic Press, London, 1996.

T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the Multiple Instance Problem with Axis-Parallel Rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

J. A. DiMasi, R. W. Hansen, and H. G. Grabowski. The price of innovation: new estimates of drug development costs. *J Health Econ*, 22(2):151–185, Mar 2003.

J. Drews. Drug Discovery: A Historical Perspective. *Science*, 287:1960–1964, March 2000.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2001.

L. M. Egolf and P. C. Jurs. Prediction of Boiling Points of Organic Hterocyclic Compounds Using Regression and Neural Networks Techniques. *J Chem Inf Comput Sci*, 33:616–635, 1993.

S. Ekins, B. Boulanger, P. W. Swaan, and M. A. Z. Hupcey. Towards a new age of virtual ADME/TOX and multidimensional drug discovery. *J Comput Aided Mol Des*, 16(5-6):381–401, 2002.

T. Fawcett. ROC graphs: notes and practical considerations for data mining researchers. Technical Report 2003-4, HP Laboratories, Palo Alto, CA, USA, 2003.

P. Finn, S. Muggleton, D. Page, and A. Srinivasan. Pharmacophore discovery using the inductive logic programming language Progol. *Machine Learning*, 30:241–270, 1998.

D. R. Flower. On the properties of bit string-based measures of chemical similarity. *J Chem Inf Comput Sci*, 38:379–386, 1998.

H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. In *Proceedings of the 22nd international conference on Machine learning*, pages 225 – 232, New York, NY, USA, 2005. ACM Press.

T. Gärtner. Exponential and Geometric Kernels for Graphs. In NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data, 2002.

T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, Heidelberg, 2003. URL `http://dx.doi.org/10.1007/b12006`.

T. Gärtner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-Instance Kernels. In C. Sammut and A. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 2002.

J. Gasteiger and T. Engel, editors. *Chemoinformatics : a Textbook*. Wiley, 2003.

V. Gillet, P. Willett, and J. Bradshaw. Similarity searching using reduced graphs. *J Chem Inf Comput Sci*, 43:338–345, 2003.

A.C. Good and W.G. Richards. Rapid Evaluation of Molecular Shape Similarity Using Gaussian Functions. *J Chem Inf Comput Sci*, 33:112–116, 1993.

M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Comput. Chem.*, 20(1):25–33, 1996.

O. F. Güner. *Pharmacophore Perception, Development, and Use in Drug Design*, volume 2 of *IUL Biotechnology Series*. International University Line, 2000.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Mach. Learn.*, 46(1/3):389–422, Jan 2002. URL `http://homepages.nyu.edu/~jaw281/genesel.pdf`.

I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins*, 47(4):409–443, Jun 2002. URL `http://dx.doi.org/10.1002/prot.10115`.

M. Hann and R. Green. Chemoinformatics–a new name for an old problem? *Curr Opin Chem Biol*, 3(4):379–383, Aug 1999. URL http://dx.doi.org/10.1016/S1367-5931(99)80057-X.

C. Hansch and T. Fujita. A method for the correlation of biological activity and chemical structure. *J. Am. Chem. Soc*, 86:1616–1626, 1964.

C. Hansch, J. E. Quinlan, and G. L. Lawrence. Linear free-energy relationship between partition coefficients and the aqueous solubility of organic liquids. *J. Org. Chem.*, 33:347 – 350, 1968.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2001.

D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999. URL http://www.cse.ucsc.edu/~haussler/convolutions.ps.

D.M. Hawkins, S.S. Young, and A. Rusinko. Analysis of a large structure-activity data set using recursive partitioning. *Quantitative Structure-Activity Relationships*, 16: 296–302, 1997.

C. Helma, T. Cramer, S. Kramer, and L. De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *J Chem Inf Comput Sci*, 44(4):1402–11, 2004. URL http://dx.doi.org/10.1021/ci034254q.

J. D. Holliday and P. Willett. Using a genetic algorithm to identify common structural features in sets of ligands. *J. Mol. Graph. Model.*, 15(4):221–232, Aug 1997.

D. Horvath and C. Jeandenans. Neighborhood behavior of in silico structural spaces with respect to in vitro activity spaces-a novel understanding of the molecular similarity principle in the context of multiple receptor binding profiles. *J Chem Inf Comput Sci*, 43:680–690, 2003. URL http://dx.doi.org/10.1021/ci025634z.

T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158–167, New York, NY, USA, 2004. ACM Press.

A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: mining graph data. *Mach. Learn.*, 50(3):321–354, 2003.

T. Jebara, R. Kondor, and A. Howard. Probability Product Kernels. *J. Mach. Learn. Res.*, 5:819–844, 2004. URL http://jmlr.csail.mit.edu/papers/v5/jebara04a.html.

T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publishers, 2002.

M. A. Johnson and G. M. Maggiora, editors. *Concepts and Applications of Molecular Similarity*. Wiley, 1990.

G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *J Mol Biol*, 267(3):727–748, Apr 1997. URL `http://dx.doi.org/10.1006/jmbi.1996.0897`.

W. L. Jorgensen. The many roles of computation in drug discovery. *Science*, 303(5665): 1813–1818, Mar 2004. URL `http://dx.doi.org/10.1126/science.1096361`.

R. N. Jorissen and M. K. Gilson. Virtual screening of molecular databases using a support vector machine. *J Chem Inf Model*, 45(3):549–61, 2005. URL `http://dx.doi.org/10.1021/ci049641u`.

Y. Karklin, R. F. Meraz, and S.R. Holbrook. Classification of non-coding RNA using graph representations of secondary structure. *Pac. Symp. Biocomput.*, pages 4–15, 2005. URL `http://helix-web.stanford.edu/psb05/karklin.pdf`.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized Kernels between Labeled Graphs. In T. Faucett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press, 2003.

H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 155–170. MIT Press, 2004.

E. Kellenberger, J. Rodrigo, P. Muller, and D. Rognan. Comparative evaluation of eight docking tools for docking and virtual screening accuracy. *Proteins*, 57(2): 225–242, Nov 2004. URL `http://dx.doi.org/10.1002/prot.20149`.

R. D. King, S. Muggleton, R. A. Lewis, and M. J. Sternberg. Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc Natl Acad Sci U S A*, 89(23):11322–11326, Dec 1992.

R. D. King, S. H. Muggleton, A. Srinivasan, and M. J. Sternberg. 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 U S A*, 93(1):438–442, Jan 1996.

D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov*, 3 (11):935–949, Nov 2004. URL `http://dx.doi.org/10.1038/nrd1549`.

G. Klebe. Recent developments in structure-based drug design. *J Mol Med*, 78(5): 269–281, 2000.

S. Kramer and L. De Raedt. Feature Construction with Version Spaces for Biochemical Applications. In C.E. Brodley and A. Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 258–265. Morgan Kaufmann, 2001.

S. Kramer, E. Frank, and C. Helma. Fragment generation and support vector machines for inducing SARs. *SAR QSAR Environ Res*, 13(5):509–23, Jul 2002. URL `http://dx.doi.org/10.1080/10629360290023340`.

H. Kubinyi. Comparative Molecular Field Analysis. In J. Gasteiger, editor, *Handbook of Chemoinformatics. From Data to Knowledge, Volume 4*, pages 1555–1574. Wiley-VCH, Weinheim, 2003.

G.R.G. Lanckriet, N. Cristianini, M.I. Jordan, and W.S. Noble. Kernel-based integration of genomic data using semidefinite programming. In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 231–259. MIT Press, 2004.

J. S. Lazo and P. Wipf. Combinatorial chemistry and contemporary pharmacology. *J Pharmacol Exp Ther*, 293(3):705–709, Jun 2000.

A. R. Leach and V. J. Gillet. *An introduction to chemoinformatics*. Kluwer Academic Publishers, 2003.

C. Lemmen and T. Lengauer. Computational methods for the structural alignment of molecules. *J. Comput. Aided. Mol. Des.*, 14(3):215–232, Mar 2000.

C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: a string kernel for SVM protein classification. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Kevin Lauerdale, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 564–575. World Scientific, 2002. URL `http://www.smi.stanford.edu/projects/helix/psb02/leslie.pdf`.

C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004. URL `http://bioinformatics.oupjournals.org/cgi/content/abstract/20/4/467`.

P. Lind and T. Maltseva. Support vector machines for the estimation of aqueous solubility. *J Chem Inf Comput Sci*, 43(6):1855–9, 2003. URL `http://dx.doi.org/10.1021/ci034107s`.

C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug. Deliv. Rev*, 46(1-3):3–26, Mar 2001.

H. X. Liu, C. X. Xue, R. S. Zhang, X. J. Yao, M. C. Liu, Z. D. Hu, and B. T. Fan. Quantitative prediction of logk of peptides in high-performance liquid chromatography based on molecular descriptors by using the heuristic method and support vector machine. *J Chem Inf Comput Sci*, 44(6):1979–86, 2004. URL `http://dx.doi.org/10.1021/ci049891a`.

Y. Liu. A comparative study on feature selection methods for drug discovery. *J Chem Inf Comput Sci*, 44(5):1823–8, 2004. URL `http://dx.doi.org/10.1021/ci049875d`.

P. Mahé, L. Ralaivola, V. Stoven, and J.-P. Vert. The pharmacophore kernel for virtual screening with support vector machines. *J Chem Inf Model*, 46(5):2003–2014, 2006. URL `http://dx.doi.org/10.1021/ci060138m`.

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J Chem Inf Model*, 45(4):939–51, 2005. URL `http://dx.doi.org/10.1021/ci050039t`.

D.T. Manallack and D.J. Livingstone. Neural networks in drug-discovery: have they lived up with their promise? *Eur. J. Med. Chem.*, 34:195–208, 1999.

C. Manly, S. Louise-May, and J. Hammer. The impact of informatics and computational chemistry on synthesis and screening. *Drug Discov. Today*, 6(21):1101–1110, Nov 2001.

Y. C. Martin, J. B. Holland, C. H. Jarboe, and N. Plotnikoff. Discriminant analysis of the relationship between physical properties and the inhibition of monoamine oxidase by aminotetralins and aminoindans. *J Med Chem*, 17(4):409–413, Apr 1974.

J. S. Mason, I. Morize, P. R. Menard, D. L. Cheney, C. Hulme, and R. F. Labaudiniere. New 4-point pharmacophore method for molecular similarity and diversity applications: overview of the method and applications, including a novel approach to the design of combinatorial libraries containing privileged substructures. *J Med Chem*, 42(17):3251–3264, Aug 1999. URL `http://dx.doi.org/10.1021/jm9806998`.

H. Matter and T. Pötter. Comparing 3D pharmacophore triplets and 2D fingerprints for selecting diverse compound subsets. *J Chem Inf Comput Sci*, 39:1211–1225, 1999.

M. J. McGregor and S. M. Muskal. Pharmacophore fingerprinting. 1. Application to QSAR and focused library design. *J Chem Inf Comput Sci*, 39(3):569–574, 1999.

M.J. McGregor and V. Pallai. Clustering of Large Databases of Compounds: Using the mdl "Keys" as Structural Descriptors. *J Chem Inf Comput Sci*, 37:443–448, 1997.

S. Miertus, G. Fassina, and P.F. Seneci. Concepts of Combinatorial Chemistry and Combinatorial Technologies. *Chemické Listy*, 94:1104–1110, 2000.

M. A. Miteva, W. H. Lee, M. O. Montes, and B. O. Villoutreix. Fast structure-based virtual ligand screening combining FRED, DOCK, and Surflex. *J Med Chem*, 48 (19):6012–6022, Sep 2005. URL http://dx.doi.org/10.1021/jm050262h.

G. Moreau and P. Broto. Autocorrelation of molecular structures: Application to SAR studies. *Nouv. J. Chim.*, 757:764, 1980.

H.L. Morgan. The Generation of Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service. *J Chem Doc*, 5:107–113, 1965.

K.-R. Müller, G. Rätsch, S. Sonnenburg, S. Mika, M. Grimm, and N. Heinrich. Classifying 'drug-likeness' with Kernel-based learning methods. *J Chem Inf Model*, 45 (2):249–53, 2005. URL http://dx.doi.org/10.1021/ci049737o.

Cath O'Driscoll. A Virtual Space Odyssey. Nature Horizon : Charting Chemical Space, 2004.

C. A. Pepperrell and P. Willett. Techniques for the calculation of three-dimensional structural similarity using inter-atomic distances. *J Comput Aided Mol Des*, 5(5): 455–474, Oct 1991.

N. Perry and V. J. van Geerestein. Database Searching on the basis of Three-Dimensional Similarity Using the sperm Program. *J Chem Inf Comput Sci*, 32: 607–616, 1992.

S. D. Pickett, J. S. Mason, and I. M. McLay. Diversity profiling and design using 3D pharmacophores : Pharmacophores-Derived Queries (PQD). *J Chem Inf Comput Sci*, 36:1214–1223, 1996.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, Sep 2005. URL http://dx.doi.org/10.1016/j.neunet.2005.07.009.

J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In T. Washio and L. De Raedt, editors, *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.

M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *J Mol Biol*, 261(3):470–489, Aug 1996. URL http://dx.doi.org/10.1006/jmbi.1996.0477.

J. Ren and D.K. Stammers. HIV reverse transcriptase structures: designing new inhibitors and understanding mechanisms of drug resistance. *Trends Pharmacol Sci*, 26:4–7, 2005.

D. Rogers and A. J. Hopfinger. Application of genetic function approximation to quantitative structure-activity relationships and quantitative structure-property relationships. *J Chem Inf Comput Sci*, 34:854–866, 1994.

G. Rücker and C. Rücker. Counts of All Walks as Atomic and Molecular Descriptors. *J Chem Inf Comput Sci*, 33:683–695, 1993.

J. Saeh, P. Lyne, B. Takasaki, and D. Cosgrove. Lead hopping using SVM and 3D pharmacophore fingerprints. *J Chem Inf Model*, 45(4):1122–1133, Jul 2005. URL `http://dx.doi.org/10.1021/ci049732r`.

N. Salim, J. Holliday, and P. Willett. Combination of fingerprint-based similarity coefficients using data fusion. *J Chem Inf Comput Sci*, 43(2):435–442, 2003. URL `http://dx.doi.org/10.1021/ci025596j`.

A.K. Saxena and P. Prathipati. Comparison of mlr, pls and ga-mlr in qsar analysis. *SAR. QSAR. Environ. Res.*, 14:433–445, 2003.

G. Schneider and P. Wrede. Artificial neural networks for computer-based molecular design. *Prog Biophys Mol Biol*, 70(3):175–222, 1998.

B . Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery & Data Mining*. AAAI Press, 1995.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002. URL `http://www.learning-with-kernels.org`.

B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, 2004.

M. Sebag and C. Rouveirol. Tractable Induction and Classification in First-Order Logic via Stochastic Matching. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 888–893. Morgan Kaufmann, 1997.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

A. Smola and R. Kondor. Kernels and Regularization on Graphs. In B. Schölkopf and M.K. Warmuth, editors, *Proceedings of 16th Annual Conference on Computational Learning Theory*, pages 144–158. Springer-Verlag, 2003.

M. J. Sorich, J. O. Miners, R. A. McKinnon, D. A. Winkler, F. R. Burden, and P. A. Smith. Comparison of linear and nonlinear classification algorithms for the prediction of drug and chemical metabolism by human UDP-glucuronosyltransferase isoforms. *J Chem Inf Comput Sci*, 43(6):2019–24, 2003. URL `http://dx.doi.org/10.1021/ci034108k`.

J. J. Sutherland, L. A. O'Brien, and D. F. Weaver. Spline-fitting with a genetic algorithm : a method for developing classification structure-activity relationships. *J Chem Inf Comput Sci*, 43:1906–1915, 2003.

S. J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21(Suppl. 1):i359–i368, Jun 2005. URL `http://dx.doi.org/10.1093/bioinformatics/bti1055`.

Y. Takaoka, Y. Endo, S. Yamanobe, H. Kakinuma, T. Okubo, Y. Shimazaki, T. Ota, S. Sumiya, and K. Yoshikawa. Development of a method for evaluating drug-likeness and ease of synthesis using a data set in which compounds are assigned scores based on chemists' intuition. *J Chem Inf Comput Sci*, 43(4):1269–75, 2003. URL `http://dx.doi.org/10.1021/ci0340431`.

M. Thimm, A. Goede, S. Hougardy, and R. Preissner. Comparison of 2D similarity and 3D superposition. Application to searching a conformational drug database. *J Chem Inf Comput Sci*, 44(5):1816–1822, 2004. URL `http://dx.doi.org/10.1021/ci049920h`.

K. Tsuda, T. Kin, and K. Asai. Marginalized Kernels for Biological Sequences. *Bioinformatics*, 18:S268–S275, 2002.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

M. Wagener, J. Sadowski, and J. Gasteiger. Autocorrelation of molecular surface properties for modeling. corticosteroid binding globulin and cytosolic. ah. receptor. activity by neural networks. *J. Am. Chem. Soc.*, 117:7769–7775, 1995.

M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmen. Active learning with support vector machines in the drug discovery process. *J Chem Inf Comput Sci*, 43(2):667–673, 2003. URL `http://dx.doi.org/10.1021/ci025620t`.

J. Weston, F. Pérez-Cruz, O. Bousquet, O. Chapelle, A. Elisseeff, and B. Schölkopf. Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, 19(6):764–771, 2003. URL `http://bioinformatics.oupjournals.org/cgi/content/abstract/19/6/764`.

P. Willett. Chemical Similarity Searching. *J Chem Inf Comput Sci*, 38:983–996, 1998.

D. Wilton, P. Willett, K. Lawson, and G. Mullier. Comparison of ranking methods for virtual screening in lead-discovery programs. *J Chem Inf Comput Sci*, 43(2):469–74, 2003. URL `http://dx.doi.org/10.1021/ci025586i`.

L. Wolf and A. Shashua. Learning over Sets using Kernel Principal Angles. *J. Mach. Learn. Res.*, 4:913–931, 2003. URL `http://jmlr.csail.mit.edu/papers/v4/wolf03a.html`.

J. Xu and A. Hagler. Chemoinformatics and Drug Discovery. *Molecules*, 7:566–600, 2002. URL `www.mdpi.org/molecules/papers/70800566.pdf`.

C. X. Xue, R. S. Zhang, H. X. Liu, M. C. Liu, Z. D. Hu, and B. T. Fan. Support vector machines-based quantitative structure-property relationship for the prediction of heat capacity. *J Chem Inf Comput Sci*, 44(4):1267–74, 2004a. URL `http://dx.doi.org/10.1021/ci049934n`.

C. X. Xue, R. S. Zhang, H. X. Liu, X. J. Yao, M. C. Liu, Z. D. Hu, and B. T. Fan. An accurate QSPR study of O-H bond dissociation energy in substituted phenols based on support vector machines. *J Chem Inf Comput Sci*, 44(2):669–77, 2004b. URL `http://dx.doi.org/10.1021/ci034248u`.

C. X. Xue, R. S. Zhang, H. X. Liu, X. J. Yao, M. C. Liu, Z. D. Hu, and B. T. Fan. QSAR models for the prediction of binding affinities to human serum albumin using the heuristic method and a support vector machine. *J Chem Inf Comput Sci*, 44(5):1693–700, 2004c. URL `http://dx.doi.org/10.1021/ci049820b`.

L. Xue and J. Bajorath. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Comb. Chem. High. Throughput Screen.*, 3(5):363–372, Oct 2000.

L. Xue, F. L. Stahura, J. W. Godden, and J. Bajorath. Fingerprint scaling increases the probability of identifying molecules with similar activity in virtual screening calculations. *J Chem Inf Comput Sci*, 41(3):746–753, 2001a.

L. Xue, F. L. Stahura, J. W. Godden, and J. Bajorath. Mini-fingerprints detect similar activity of receptor ligands previously recognized only by three-dimensional pharmacophore-based methods. *J Chem Inf Comput Sci*, 41(2):394–401, 2001b.

Y. Xue, Z. R. Li, C. W. Yap, L. Z. Sun, X. Chen, and Y. Z. Chen. Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents. *J Chem Inf Comput Sci*, 44(5):1630–8, 2004d. URL `http://dx.doi.org/10.1021/ci049869h`.

Y. Xue, C. W. Yap, L. Z. Sun, Z. W. Cao, J. F. Wang, and Y. Z. Chen. Prediction of P-glycoprotein substrates by a support vector machine approach. *J Chem Inf Comput Sci*, 44(4):1497–505, 2004e. URL `http://dx.doi.org/10.1021/ci049971e`.

X. J. Yao, A. Panaye, J. P. Doucet, R. S. Zhang, H. F. Chen, M. C. Liu, Z. D. Hu, and B. T. Fan. Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. *J Chem Inf Comput Sci*, 44(4):1257–66, 2004. URL `http://dx.doi.org/10.1021/ci049965i`.

C. W. Yap and Y. Z. Chen. Prediction of Cytochrome P450 3A4, 2D6, and 2C9 Inhibitors and Substrates by Using Support Vector Machines. *J Chem Inf Model*, 45(4):982–92, 2005. URL `http://dx.doi.org/10.1021/ci0500536`.

V. V. Zernov, K. V. Balakin, A. A. Ivaschenko, N. P. Savchuk, and I. V. Pletnev. Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions. *J Chem Inf Comput Sci*, 43(6):2048–56, 2003. URL `http://dx.doi.org/10.1021/ci0340916`.

J. Zupan and J. Gasteiger. *Neural Networks in Chemistry and Drug Design*. Wiley-VCH, 1999.