KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Computer Science

Predictive Quantitative Structure-Activity Relationship Models and their use for the Efficient Screening of Molecules

ir. Kurt DE GRAVE

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

30 August 2011

Predictive Quantitative Structure-Activity Relationship Models and their use for the Efficient Screening of Molecules

ir. Kurt DE GRAVE

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering

30 August 2011

D/2011/7515/99
ISBN 978-94-6018-397-3

Contents

List of Figures

List of Algorithms

List of Tables

List of Symbols

$[a]$	Concentration of a or singleton vector containing the number a
$[a\ b]$	Row vector
G	Graph
v, u	Vertex
e	Edge
uv	Edge joining u and v
V_G	Set of vertices of graph G
E_G	Set of edges of graph G
G_v	Graph rooted in v
$d(u, v)$	Topological distance between u and v
$\mathcal{G}$	Set of simple connected graphs
$N_r(v)$	Neighborhood of radius r and of vertex v
Ω_r^v	Neighborhood subgraph of radius r and of vertex v
$\lambda(v)$	Label of vertex v
$\lambda(uv)$	Label of edge uv
Σ	Alphabet (of the label function)
$\simeq$	Is isomorphic to
$\wedge$	And
K	Kernel
κ	Kernel over parts
$\langle x, y\rangle$	Scalar product between x and y
ϕ	Feature map
Φ	Feature matrix
$^\top$	Transpose
w	Weight vector
x_i	ith training instance or set of input facts of ith interpretation
y_i	Target value of ith training instance or set of output facts of ith interpretation
b	offset of a linear function or level of optimism
ξ_k	Slack variable for the kth constraint

c	Trade-off between training errors and model complexity
G_{aug}	Augmented molecular graph
δ	Kronecker delta
R	Parts-of relation
$\mathbb{N}$	Set of natural numbers (including zero)
$\mathbb{R}$	Set of real numbers
$\lvert \cdot \rvert$	Cardinality of a set or absolute value of a number or determinant of a matrix
$\lVert \cdot \rVert_2$	Euclidean vector norm
$\mathbb{E}(X), \bar{X}$	Expectation of random variable X
$O(\cdot)$	Big O notation
r^*	Maximum radius (of neighborhood subgraphs)
d^*	Maximum distance (between the roots of neighborhood subgraphs)
z	Interpretation
$\mathcal{C}$	Set of domain constants
$\mathcal{R}$	Set of relations
r/m	Relation r of arity m
$\mathcal{E}$	Set of entity identifiers
$\mathcal{E}_i$	ith entity set
$\mathcal{V}$	Set of property values
$\lambda_t(v)$	t-th component of the label tuple associated to v
$\lambda_p(v)$	predicate name associated to v
$\mathcal{P}$	Pool of instances
N_{max}	Budget for experimentation
$\lVert T \rVert_{\text{best}-k}$	Arithmetic mean of the k largest elements of vector T
$\mathcal{N}(m, \sigma^2)$	Gaussian distribution with mean m and standard deviation σ
$\mathcal{N}(M, \Sigma)$	Multi-variate Gaussian distribution with mean vector M and covariance matrix Σ
σ	Standard deviation of noise
$\sim$	Is distributed as
$\propto$	Is proportional to
X_N	Vector of the first N examples
T_N	Target value vector of the first N examples
Φ_N	Feature matrix of the first N examples
I	Identity matrix
$P(x\|y)$, $x\|y$	Conditional probability of x given y
$t_{\#(k,N)}$	Target value of the kth best example at step N
$\binom{n}{k}$	Choose k from n (binomial coefficient)
wt%	Mass (weight) percent
$\lfloor x \rfloor$	Largest integer not bigger than x (floor)

Abstract

We explore two avenues where machine learning can help drug discovery: predictive models of in vivo or in vitro effects of molecules (known as Quantitative Structure-Activity Relationship or QSAR models), and the selection of efficient experiments based on such models.

In the first part, we present methods to improve the predictive power of graph kernel based molecule classifiers. The bias of existing graph kernels can be improved by augmenting atom-bond graphs with functional groups. This novel representation allows a machine learning algorithm to use both high-level functional and low-level atomic information, without any change to the kernel or learning algorithm. In internal validation tests, we observe consistently higher AUROCs for all tested kernels.

We also introduce a novel, efficient graph kernel called the Neighborhood Subgraph Pairwise Distance Kernel. The feature space of this kernel is the space of pairs of topological balls and the interpair distance. Using this kernel, a standard support vector machine outperforms existing methods in the prediction of all investigated target properties: mutagenicity, in vivo toxicity, antiviral activity, and cancer suppression.

In the second part, we tackle the problem of efficient experimentation in drug discovery using optimization assisted by a learned surrogate model and we evaluate different experiment selection strategies. The algorithm is extended to accommodate drug discovery needs, such as the selection of many parallel experiments. The algorithm is integrated in an automated drug discovery platform, the robot scientist Eve. It is also applied to the optimization of the design of nanofiltration membranes.

Beknopte samenvatting (Abstract in Dutch)

In dit werk bestuderen we twee manieren om het onderzoek naar geneesmiddelen efficiënter en effectiever te maken.

In het eerste deel stellen we methoden voor om voorspellende modellen van in-vivo of in-vitro effecten van moleculen nauwkeuriger te maken. De nauwkeurigheid van modellen gebaseerd op grafenkernels kan worden verbeterd door functionele groepen op te nemen in de graaf, die normaal enkel uit atomen en hun bindingen bestaat. Een leeralgoritme kan dan zowel rekening houden met grotere functionele eigenschappen als met de exacte structuur van de individuele atomen, zonder dat we het leeralgoritme of de kernel moeten wijzigen. We stellen experimenteel vast dat alle geteste eigenschappen (toxiciteit, antivirale werking, en remming van kankergroei) nauwkeuriger worden voorspeld als de uitgebreide grafen worden gebruikt.

We introduceren ook een nieuwe, efficiënte kernel, die we de Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) noemen. De NPSDK vergelijkt moleculen door te tellen hoeveel paren van topologische bollen er in hun graaf te vinden zijn. Hoe meer dergelijke paren identiek zijn in twee moleculen en bovendien dezelfde interpaar afstand hebben, hoe gelijkaardiger de moleculen worden bevonden. Met behulp van de NSPDK levert een gewone Support Vector Machine nauwkeurigere voorspellingen van de toxische, antivirale en antikanker eigenschappen van moleculen dan welke andere methode ook.

In het tweede deel bestuderen we hoe experimenten kunnen worden geselecteerd. Omdat er een enorm aantal verschillende moleculen kunnen worden gesynthetiseerd (waarvan vele miljoenen commericieel beschikbaar zijn), is de vraag welke moleculen het eerst te testen, van groot belang. Dit geldt des te meer naarmate de test duurder is. Slechts een minieme fractie van alle moleculen heeft immers een gewenst geneeskundig effect, dus is het verre van denkbeeldig dat het budget op raakt voor een geschikte molecule gevonden wordt. We passen surrogaatmodel-gebaseerde optimalisatietechnieken toe op dit probleem en we vergelijken manieren om de moleculen te kiezen die moeten worden getest. We breiden het algoritme uit zodat het vele moleculen tegelijk kan selecteren in plaats van één-per-één. De toepasbaarheid van het algorithme wordt bewezen door de integratie in een automatische screening-robot. We passen het algoritme ook toe om te zoeken naar betere membranen voor ultrafiltratie.

Acknowledgments

A doctoral project is a major undertaking and during these years, many people got to deserve my gratitude. I will only mention a small subset of them. There are surely many more people who deserve to be mentioned here, be it for their scientific contribution or their moral support and enjoyable company. If you are wrongfully withheld proper acknowledgment, please forgive me.

First, I want to thank my promotors, Luc De Raedt and Jan Ramon. I was lucky to have both a great visionary in all matters of machine learning, and a great math wizard as promotor. Jan is confident, pragmatic, and he seems to be able to solve any riddle. Being co-located with him was a great privilege, for Jan is always willing to discuss any ideas and issues. Luc seems to know everything ever published on relational learning. He learned me how science works and how to write papers. He is also very supportive when things don't go as hoped for. I deeply appreciate the promotorship of these great men.

Much of my research in the last two years was in collaboration with Fabrizio Costa, the man who has a splendid new idea every day. Fabrizio would drop by to discuss his daily idea, and together we'd find out if it was practicable or not, and try to imagine an alternative that was, often for hours on end. Fabrizio is a researcher pur sang. Fabrizio, thanks, it has been very enjoyable to work with you.

Paolo Frasconi arrived in Leuven late in my Ph.D. career, but nevertheless had a lot of influence. Paolo is a very good person; he would never complain even when I was disappointed with my own work. Paolo, thanks for sharing your deep insights.

Furthermore, I would like to thank Ross King and Michael Berthold for inviting me to visit their lab. I also want to thank the members of my evaluation committee (of which Maurice Bruynooghe, Yves Moreau, and chair Ann Haegemans have not yet been named) for their time and their constructive comments.

Many people contributed to the good atmosphere at DTAI. Leander, Celine Jelle, Fabian, and Christophe were great office mates. Angelika, Anton, Bogdan, Guy, Siegfried, Tias, and others made lunch breaks most enjoyable. Leander and Tias also helped me considerably with my research. Thank you all.

Who convinced me pursue a Ph.D., years after leaving the university for industry? During the final year of my Master's degree, Hendrik Blockeel got me (even more) interested in machine learning. His inspiring teaching played an important role in the professional choices I've made ever since. Another crucial factor was the example set by Jan Struyf, one of my best friends for many years now. After we'd written a joint Master's thesis, he took up the offer to start a Ph.D. while I joined the startup PharmaDM (I got the chance thanks to Hendrik). We kept in contact, so I learnt about what it's like to do a Ph.D., and I got to know several other doctoral students, further lowering the treshold. One other close friend, Karel Van Oudheusden, also finished a Ph.D. We've had discussions for many hours about esoteric subjects often related to Gödel's incompleteness. The colleagues at PharmaDM also played an important role, especially Luc Dehaspe and Henk Vandecasteele. Luc is an optimist who believes in people, giving them the opportunity to learn. I only came to fully appreciate the Unix philosophy, FLOSS, and Prolog through Henk. My brother Koen had also shown repeatedly that a Master's degree doesn't necessarily end one's education. Thank you all for giving me the motivation to start and persist.

A lot of people have provided essential support, without which research would have been much more difficult. In particular, the help of Karin Michiels, the team of dedicated sysadmins at Computer Science and the HPC team at ICTS was invaluable.

I am grateful to the sponsors of my work: Bijzonder Onderzoeksfonds (GOA/08/008 "Probabilistic Logic Learning"), the institute for Innovatie door Wetenschap en Technologie (Galapagos O&O project on antivirals), and the European Research Council (Starting Grant 240186 "MiGraNT: Mining Graphs and Networks, a Theory-based approach"). I'm also grateful to Johan Neyts for giving me a lot of flexibility for finishing up this text while working at his lab.

Finally, I want to thank the people who helped me most of all, my parents. They are dedicated to their children to a degree I've not seen in other families. They've done everything they could to give us better opportunities in life than they'd gotten themselves.

Outline

No sensible decision can be made any longer
without taking into account
not only the world as it is,
but the world as it will be.

Isaac Asimov

Introduction

The goal of this thesis is to develop computational methods to make drug discovery, and functional chemistry research in general, more efficient and effective.

Drug discovery is an empirical science. The primary tool for identifying a molecule that can become a prescription medicine, is to create a simplified, reproducible biological model of the disease at the cell level or at the molecular level. The model becomes an assay when it is designed to allow convenient detection whether the disease is still functional after some manipulation. Fluorescent marker molecules are often the detection method of choice, but a myriad of other mechanisms are also used. To test a molecule it is just added to the assay solution. After an appropriate time, which can be milliseconds or days, the presence or absence of the marker is observed. If the disease is inhibited, we say that the molecule is active.

Realising that drug discovery is experiment driven, we will consider three questions in particular with regard to our stated goal.

Q1 Can we generalize the results of experiments in order to predict the activity of molecules that have not been tested?

Question Q1 asks, once a certain number of experiments have been performed, whether we can use the results to predict for other molecules how well they would perform in the assay. This comprises the automatic construction of accurate computer models for predicting an arbitrary property of small molecules. Once created, the input for the model is just the chemical description of the molecule — how it is composed of atoms of the different elements and how its atoms are bonded together. The output can be a real number, describing the expected magnitude of activity in the assay. Alternatively, the output may describe the probability that the molecule is active, or it may just be a Boolean telling us whether the model expects the molecule to be active or inactive. All such models are called Quantitative Structure-Activity Relationships (QSAR), and they establish the correlation between a molecule's structure and its activity in a particular assay.

If the property we want to predict is not activity against a disease, but rather lipophilicity or toxicity, the more general term Quantitative Structure-Property Relationship (QSPR) is more appropriate. However, the term QSAR has been in use longer and is also frequently used in the more general sense. We will use both terms interchangeably.

The purpose of a QSAR model can be twofold:

Predictive To displace or reduce the need for actually measuring the compounds in the assay, thereby saving resources.

Descriptive To explain to the researcher what properties are important in a molecule for it to be an effective candidate drug.

Usually there is a trade-off between the intelligibility or transparency of a model and the quality of its predictions. That is, the most accurate models do not allow one to pinpoint concisely what aspects or features matter in a molecule. Typically, in a biological system many aspects are important at the same time, even though to different degrees. As is evident from the title of this thesis, we will concentrate on models that are predictive rather than descriptive.

To create the QSAR model, we will assume that some molecules have already been tested in the assay, and we will exploit these measurements (the 'training data') as well as we can. However, that does not mean we should adjust the parameters of the model such that the model perfectly fits all available data. Real-world measurements produce noisy data. That is, the measured values do not reflect the underlying phenomenon exactly but contain random errors. This may be due to the limited precision of the measurement method, or because there are uncontrolled variables other than the molecular structure that contribute to the outcome. Theory and experience show that in the presence of noise (and few fields can compete with biology when it comes to producing noisy data), it is better for the model to remain sufficiently 'simple' than to cover all subtleties apparent in the training data. The model will then only pick up the patterns that have suffient support in the training data and are therefore hopefully also present and valid in the test data. The capability to accurately predict not just the training data but also unseen test data, is called *generalization*. The scientific discipline which studies methods to build such models, is *machine learning*.

Intuitively, we can expect our models to be better when during their construction we have access to more training data. Indeed, this is a key property of a machine learning algorithm. Mitchell (Mitchell 1997) defines machine learning as the capability of a computer program to improve its performance with experience. The improvement can be measured on different scales (accuracy, mean squared error, ...) and the precise task may differ, but the defining capability is to obtain some benefit from experience.

In drug discovery the 'experience' is the data on the molecules that have already been tested. They are the *training instances* or training examples. The task of building a model based on training data is called *supervised learning*, as opposed to unsupervised learning where one does not have access to any training labels (e.g. clustering). If the target property is discrete (e.g. active vs. inactive), the machine learning problem is called *classification*. If the target is a real number (e.g. EC_{50}, the log-concentration at which the compound achieves 50% effect), the problem is called *regression*. We will initially focus on the learning task known as *binary classification*: predicting whether a molecule has a particular property or not.

When the predictions of a model are used as a substitute for screening new molecules in the original assay, the process is called *virtual screening*. Replacing expensive wet lab tests with cheap *in silico* computations is obviously desirable, if the model is accurate enough. However, in the pursuit of making drug discovery and functional chemistry more efficient, the ability to build accurate predictive models is only a first step — or rather the last step. Indeed, at the start of each research project, there is not enough data to build a sufficiently accurate model. The screening of molecules in an assay is a process during which critical decisions must be made on which molecules to screen, and how many, rather than a static provider of input data. Chemical space, the set of molecules that can possibly be synthesised, is extremely large. A rough estimate in (Bohacek et al. 1996, page 43) arrives at 10^{63} organic molecules of up to 30 atoms. A precise number is very difficult to obtain and depends strongly on the assumptions one makes concerning size, stability, synthetic accessibility, permitted elements, and so on. No matter the precise size of chemical space, it is clearly infeasible to screen all of it. Even if it was, the drug discovery researcher is not necessarily interested in the activity of all possible compounds. She is rather interested in finding a sufficient number of very active compounds. The inactive compounds are of no interest at all. This brings us to the second question:

Q2 Can we design experiments such that we need to carry out fewer of them?

Considering the end goal of the drug discovery researcher, we will reinterprete (actual, not virtual) screening as an iterative process in which compounds (or batches of many compounds) are tested one after another rather than all at the same time. We will assume that the researcher has some form of opportunity to choose the next compounds to be tested, even though there may be constraints. The task is then to iteratively select the best set of compounds. It is a machine learning task, for we expect such an algorithm to gradually become better at picking new compounds as it obtains more evidence. It is, however, an unusual task in machine learning. Under the same conditions, we can also abstract the activity of the drug discovery researcher as a function optimization process. We can then apply methods from the field of global optimization. The function to optimize is very expensive compared to in silico computations, since its evaluation consists of perfoming a real-world assay experiment, thereby consuming resources and compound stock. We will therefore apply optimization methods specialized for requiring as few function evaluations as possible. The most powerful of such methods involve the construction of a predictive model of the function to be optimized after each batch of function evaluations. The model is a surrogate: it can be queried cheaply in silico instead of evaluating the real function, which involves using the assay. Learning the surrogate function corresponds to building a QSAR model. However, surrogate-based optimization works better if we also obtain a good estimate of how accurate the model is in different regions of its domain. For that reason, we will use different QSAR modeling techniques in answering Question Q2 than for Q1. Answering Q2 also demands the investigation of different optimization heuristics that make use of the QSAR model. We are especially interested in their relative frugality: how few experiments they require to find comparably good compounds.

Answering Q1 and Q2 will provide us with a set of advanced tools. A final question we should then ask, is:

Q3 Are any of the insights and techniques obtained while answering Q1–Q2 relevant for domains outside drug discovery?

In our quest to answer Q1, we will design novel predictive methods for molecules. More precisely, we will propose (among others) a new kernel for molecular graphs[1] that leads to more accurate predictions. The new kernel also works on other types of graphs, and we will also investigate how it can be applied to other types of data. This will lead to a new *language* for machine learning. Using the new language, kernel methods can be applied to the most general type of data in machine learning: relational data.

For the global optimization methods investigated while seeking an answer for Q2, we will also test an application outside drug discovery, even though without straying from the field of functional chemistry: we will apply the method to the design of water filtration membranes.

Contributions and Roadmap

The thesis is organized into two main parts. The first part discusses machine learning models that predict the activity of molecules in a biological assay based on the structural formulae. The second part describes how to use such models in a setting where one wants to obtain highly active molecules for a given experimental budget.

Part I covers the construction of empirical models aimed at accurately predicting the activity of ligand molecules based on past screening data of other ligands — cf. Question Q1 in the introduction.

First, Chapter 1 concisely introduces the prerequisite mathematics of graphs and kernels. The chapter contains no original contribution by the author; its sole function is to introduce the definitions and notation that will be used throughout Part I. Some concepts will also recur in Part II. Readers knowledgeable in both mathematical domains may skip the chapter.

Chapter 2 introduces a new representation of molecules, which embeds chemical background knowledge into the traditional atom-bond representation. The supplementary knowledge improves the bias of graph-based machine learning algorithms.

Chapter 3 introduces the neighborhood subgraph pairwise distance kernel (NSPDK). When this novel molecule kernel is fed into a standard support vector machine (SVM) algorithm, a state-of-the art ligand-based activity prediction system emerges, as observed with extensive in silico experiments on datasets ranging from toxicology over oncology to virology.

[1]What kernels and graphs are, will become clear in Chapter 1.

NSPDK has inspired the conception of a generic kernel-based machine learning system that accepts any relational database as input, not just molecular structures. This system, called kLog, operates (conceptually) in three steps. First, the relational data is graphicalized. A graph kernel or feature generator is then applied, and finally a model is learned from the kernel values or features. We describe kLog in Chapter 4. It provides a first part of our answer to Question Q3.

Part I is based on material that has appeared in the following publications (key articles only):

> De Grave, K. and Costa, F. (2010). *Molecular graph augmentation with rings and functional groups.* Journal of Chemical Information and Modeling.
>
> Costa, F. and De Grave, K. (2010). *Fast neighborhood subgraph pairwise distance kernel.* In: Proceedings of the 26th International Conference on Machine Learning.
>
> Frasconi, P. and Costa, F. and De Raedt, L. and De Grave, K. (2011). *kLog: a language for logical and relational learning with kernels*, Technical report, Katholieke Universiteit Leuven, Università degli Studi di Firenze, and Albert-Ludwigs-Universität.

Part II covers screening methodologies for expensive assays — cf. Question Q2. We assume that the screening occurs incrementally rather than all at once. The core idea is to support the selection of experiments (new molecules to be screened) as much as possible with information obtained from already screened molecules. The screening design is thus interpreted as a function optimization problem.

Chapter 5 describes this setting in detail and identifies a selection of algorithms which are suitable for screening candidate selection. The algorithms are compared in silico against each other and against traditional, non-incremental, high throughput screening. It is shown that drug discovery researchers with finite budgets should consider using an online optimization algorithm to run their screenings. In particular, the 'optimistic' selection of candidates algorithm is a good choice, even though it lacks the desirable theoretical property of dense sampling. The algorithm is implemented and being tested in the wet lab practice by the drug discovery robot scientist project at the University of Aberystwyth. This high-profile project aims at the development of Eve, an autonomous robotic system to demonstrate the end-to-end automation of scientific discovery in the field of drug discovery. Eve is described in Chapter 6.

Chapter 7 shows that the applicability of optimization algorithms is not restricted to the small-molecule ligand area of the chemical space. They can just as well be applied to the other extreme of the weight scale of organic molecules: polymers. In particular, we consider the problem of the synthesis of cellulose acetate nanofiltration membranes. This chapter completes our answer to Question Q3.

Part II is based partly on material that has appeared in the following publications:

> De Grave, K and Ramon, J. and De Raedt, L. (2008). *Active learning for high throughput screening.* International Conference on Discovery Science. Lecture Notes in Computer Science 5255, 185–196. *Received the 2008 Carl Smith Award.*
>
> Cano Odena, A. and Spilliers, M. and Dedroog, T. and De Grave, K. and Ramon, J. and Vankelecom, I. F. J. (2010) *Optimization of Cellulose Acetate Nanofiltration Membranes for Micropollutant Removal via Genetic Algorithms and High Throughput Experimentation.* Journal of Membrane Science.
>
> Cano Odena, A. and Vandezande, P. and Cools I. and Vanderschoot, K. and De Grave, K. and Ramon, J. and De Raedt, L. and Vankelecom I. F. J. (2008) *Comparison of Multi-Parameter Optimization Strategies for the Development of Nanofiltration Membranes for Salt and Micropollutants Removal.* In: International Congress on Membranes and Membrane Processes (ICOM 2008).

The above list again contains only the most relevant and important articles. A full publication list of the author can be found in appendix.

Part III summarizes the findings of this thesis and indicates future research directions.

Part I

Quantitative Structure-Activity Relationships

Chapter 1

Molecules, graphs, and kernels

To successfully build models of a molecular property, we need to represent molecules, and relations between molecules, mathematically. Two types of mathematical objects will dominate the following chapters: graphs and kernels. In this chapter, we introduce some prerequisites.

1.1 QSAR

A Quantitative Structure-Property Relationship (QSPR) is the relation between a specific empirical property of chemical compounds and their chemical structures. Often the empirical property under consideration is the activity of the compound in a biological assay, e.g. anti-viral effect or growth suppression of a cancer cell culture. In that case, the term Quantitative Structure-Activity Relationship (QSAR) is used (Dudek et al. 2006). The latter term is older and often used as a synonym for QSPR. The chemical structure of a compound is its composition of atoms of different elements and the way in which its atoms are bonded together. We will use the symbol G for a chemical structure, for reasons that will become clear in the next section, and even clearer in the next chapter. The most general form of QSAR can be described by the formula

$$log([A_G \rightleftharpoons B_G]) = f(G) \tag{1.1.1}$$

or: the log-concentration at which there is equilibrium between two chemical states A_G and B_G is a function of the structure of the molecule. The task of the QSAR modeler is to find a function f that closely approximates reality. To have any hope of achieving this, (s)he has to assume that the activity is not random, but that similar structures will show similar activities.

In the current part of the thesis, I will focus on binary QSARs[1], where I am only interested in discriminating between molecules that are active below a threshold concentration C, and molecules that are not:

$$f(G) = \begin{cases} +1 & \text{if } [A_G \rightleftharpoons B_G] \leqslant C \\ -1 & \text{otherwise} \end{cases} \tag{1.1.2}$$

Often only a single concentration is measured, in which case we have to model presence or absense of effect at that concentration.

A seminal work on QSAR is (Hansch et al. 1962). The term 'Hansch analysis' now refers to a regression model where the biological activity is expressed as a linear or parabolic combination of few physico-chemical properties. The logarithm of the octanol-water partition coefficient[2] is one of the empirical properties often used. It can be approximated by its own QSPR model if no experimental results are available.

Free and Wilson in another important early work (Free and Wilson 1964) studied the activity of a series of congeneric compounds[3] and found an independent and additive contribution to activity at each substituent location. They consequently express activity as a linear combination:

$$log([A_G \rightleftharpoons B_G]) = \sum_{ij} a_{ij} x_{ij} + \mu \tag{1.1.3}$$

where x_{ij} indicates whether substituent i is present in position j on the scaffold (one is present at each position, i.e. $\sum_j x_{ij} = 1$), and a_{ij} is the strength of the contribution.

Free-Wilson analysis, and in most cases Hansch analysis as well, is limited to congeneric compounds. To construct a more sophisticated function f, we need a way to represent structures in the computer. The most common and natural representation is by means of a labeled graph.

[1]Some use the term SAR rather than QSAR if the activity is Boolean. Here, the adjective 'quantitative' is warranted since the weights in our models will be numerical, and the binary predictions of the models will be accompanied by a numerical measure of confidence (the distance from the hyperplane, see Section 1.4).

[2]The octanol-water partition coefficient or lipophilicity is the ratio of concentration of the compound in n-octanol versus in water in a flask containing both solvents.

[3]Congeneric means that the molecules are very similar. They have the same scaffold (backbone structure), with only a few substituent groups distinguishing the individual compounds.

1.2 Graph definitions and notation

We mostly follow the notation in (Gross and Yellen 2003).

Definition 1.2.1. Graph, vertex, and edge *A graph $G = (V, E)$ consists of two sets V and E. The elements of V are called* vertices *or nodes and the elements of E are called* edges*. Each edge has a set of two elements in V associated with it, which are called its* endpoints*. An edge is said to* join *its endpoints. We also say that an edge is* incident *to both of its endpoints.*

The notation V_G and E_G is used when G is not the only graph being considered. When G bears a subscript, we may omit the symbol G itself, e.g. $E_{(a)}$ may substitute for $E_{G_{(a)}}$. We denote an edge by its own variable (usually e), or by concatenating the variables of the endpoints, e.g. we represent the edge between the vertices u and v with uv. We will only consider undirected edges, that is, we do not distinguish between the endpoints of an edge: $uv \equiv vu$.

Figure 1.1a shows an example graph:

$$G_{(a)} = \left(V_{G_{(a)}}, E_{G_{(a)}}\right) = (\{v_1, v_2, v_3, v_4, v_5\}, \{v_1v_2, v_1v_3, v_2v_3, v_3v_4, v_4v_5\})$$

Definition 1.2.2. Adjacent vertex *A vertex v is* adjacent *to a vertex u iff they are joined by an edge.*

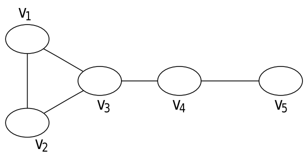
Definition 1.2.3. Vertex degree *The* degree *of a vertex is number of edges incident to it. We will denote it by the unary function* $\text{degree}(v)$.

For example, in Figure 1.1a, $\text{degree}(v_5) = 1$, $\text{degree}(v_1) = \text{degree}(v_2) = \text{degree}(v_4) = 2$, and $\text{degree}(v_3) = 3$.

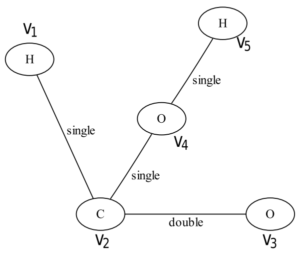
Definition 1.2.4. Simple graph *A multi-edge is a collection of two or more edges having identical endpoints. A self-loop is an edge that joins a single endpoint to itself. A* simple graph *is a graph that has no self-loops nor multi-edges.*

Figure 1.1c shows a graph that is not simple because it contains a self-loop and a multi-edge. In this thesis we consider only simple graphs. Multi-edges, which are potentially useful for the representation of multiple covalent bonds in molecules, will be substituted for by labels, to be introduced below (Definition 1.2.14). As a consequence, in Figure 1.1b, the carbon has degree three rather than four.

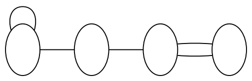
Definition 1.2.5. Complete graph *A graph is* complete *iff every pair of vertices is joined by an edge.*



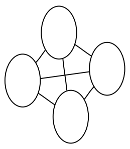
(a) Unlabeled graph.



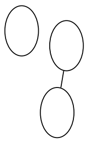
(b) Labeled graph representing the molecule formic acid.



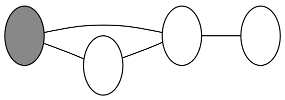
(c) Graph that is not simple.



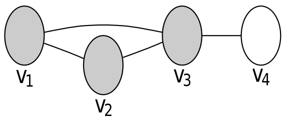
(d) Complete graph.



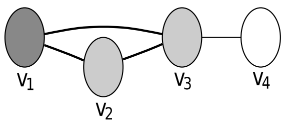
(e) Disconnected graph.



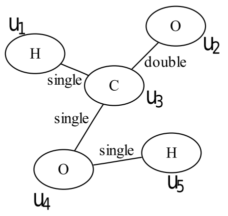
(f) Rooted graph.



(g) Neighborbood $N_1(v_1) = \{v_1, v_2, v_3\}$



(h) Neighborbood subgraph $\Omega_1^{v_1} = (\{v_1, v_2, v_3\}, \{v_1v_2, v_2v_3, v_1v_3\})_{v_1}$



(i) A labeled graph that is isomorphic to graph (b).

Figure 1.1: Example graphs.

Definition 1.2.6. Rooted graph *A graph is* rooted *when we distinguish one of its vertices, called the* root.

We denote a rooted graph G with root vertex v with G_v.

When comparing graphs, it is useful to consider substructures that are larger than individual vertices or edges. The simplest substructures we discern are linear in nature: walks and paths.

Definition 1.2.7. Walk *A* walk *in a graph G is a sequence of vertices $W = v_0, v_1, \ldots, v_n$ such that for $j = 1, \ldots, n$, the vertices v_{j-1} and v_j are adjacent. The* length *of a walk is the number of edges (counting repetitions).*

Definition 1.2.8. Path *A* path *is a walk such that no vertex is repeated, except at most the initial (v_0) and the final (v_n) vertex (in this case it is called a* cycle*).*

In the context of molecules, a cycle is more commonly called a *ring*.

Definition 1.2.9. Topological distance *The* topological distance *between two vertices, denoted $d(u, v)$, is the length of the shortest path(s) between them.*

Since we will not use any other distance measure between the vertices of a single graph, we can without ambiguity omit the adjective 'topological'.

In Figure 1.1a, we see that the greatest distance between any two vertices is three.

Definition 1.2.10. Connected graph *A graph is* connected *if there exists a walk between each pair of vertices.*

See Figure 1.1e for an example of a disconnected graph. We denote the class of simple connected graphs with $\mathcal{G}$.

As we will discover in Chapter 2 and to a greater extent in Chapter 3, it will be advantageous to consider neighborhood subgraphs, a class of compact substructures that are very different from linear sequences such as paths and walks.

Definition 1.2.11. Neighborhood *The* neighborhood of radius *r of a vertex v is the set of vertices at a topological distance less than or equal to r from v and is denoted by $N_r(v)$.*

An example neighborhood is shown in Figure 1.1g.

Definition 1.2.12. Induced subgraph *In a graph G, the* induced subgraph *on a set of vertices $W = \{w_1, \ldots, w_k\}$ is a graph that has W as its vertex set and contains every edge of G whose endpoints are in W.*

Definition 1.2.13. Neighborhood subgraph *The* neighborhood subgraph *of radius r of vertex v is the subgraph induced by the neighborhood of radius r of v and rooted in v. It is denoted by Ω_r^v.*

Figure 1.1h depicts an example neighborhood subgraph.

To represent molecules, we need to label vertices and edges to indicate the types of the corresponding atoms and bonds. An example of a labeled graph representing the molecule formic acid is depicted in Figure 1.1b.

Definition 1.2.14. Labeled graph *A* labeled graph *is a graph whose vertices and/or edges are labeled, possibly with repetitions, using symbols from a finite alphabet. We denote the function that maps the vertex/edge to the label as $\lambda : (V \cup E) \to \Sigma$.*

Consider Figures 1.1i and 1.1b. Barring the identity of the vertices and edges, none of the mathematical properties are different between the two graphs. Without using the identities, we cannot distinguish between them. This is described more formally by an isomorphism:

Definition 1.2.15. Isomorphism of labeled graphs *An* isomorphism *between two simple labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a bijection $\phi : V_1 \to V_2$ that preserves adjacency, i.e.*

$$\forall u, v \in V_1 : uv \in E_1 \Leftrightarrow \phi(u)\phi(v) \in E_2,$$

and that also preserves the label information, i.e.

$$\forall v \in V : \lambda(\phi(v)) = \lambda(v) \quad \wedge \quad \forall uv \in E : \lambda(\phi(uv)) = \lambda(uv).$$

When there exists an isomorphism between G_1 and G_2, we say that they are isomorphic*, or $G_1 \simeq G_2$.*

Figure 1.1i shows a graph that is isomorphic to 1.1b. It has no structural properties that differ from those of 1.1b, in other words the graphs are structurally indistinguishable. The isomorphism is

$$\phi : V_{(b)} \to V_{(i)} : \begin{cases} \phi(v_1) = u_1 \\ \phi(v_2) = u_3 \\ \phi(v_3) = u_2 \\ \phi(v_4) = u_4 \\ \phi(v_5) = u_5 \end{cases}$$

Definition 1.2.16. Graph invariant *An* isomorphism invariant *or* graph invariant *is a graph function $\psi : \mathcal{G} \to S$ for which holds that*

$$\forall G_1, G_2 \in \mathcal{G} : G_1 \simeq G_2 \Rightarrow \psi(G_1) = \psi(G_2)$$

The codomain S can be an arbitrary set, but we prefer sets that are structurally simpler than graphs, such as the set of natural numbers $\mathbb{N}$. The equality of two images $\psi(G_1)$ and $\psi(G_2)$ can then be verified trivially. If it doesn't hold, we know that the graphs are not isomorphic.

If a graph invariant never maps two non-isomorphic graphs to the same image, we call it an isomorphism certificate.

Definition 1.2.17. Isomorphism certificate *A* certificate for isomorphism *is an isomorphism invariant $\psi : \mathcal{G} \to S$ such that*

$$\forall G_1, G_2 \in \mathcal{G} : G_1 \simeq G_2 \Leftrightarrow \psi(G_1) = \psi(G_2)$$

Some graph invariants are very cheap to compute, e.g. the number of vertices or edges. However, there is no known isomorphism certificate algorithm for $\mathcal{G}$ with polynomial time complexity, nor is it known whether one exists.

1.3 Kernel methods

Kernel methods have been proved to achieve state-of-the-art performance in many machine learning tasks. Due to their versatility they can be employed in domains where the instances are most conveniently represented in a structured form, such as sequences, and especially important to us, graphs. This property makes them an excellent choice to tackle machine learning tasks in bioinformatics and chemoinformatics, where nucleic or amino acid sequences are naturally represented as linear chains, and molecules are represented as graphs.

The main idea in the kernel approach is to devise a computationally efficient way to calculate the similarity between two instances. If the similarity measure possesses certain qualities as described below, the task of building a specific class of machine learning models can be cast into a convex optimization problem. Convexity guarantees the existence and computability of a globally optimal solution[4], that is, there is no risk to obtain approximate models that have just reached local optima, as in the case of neural network techniques. To solve the convex optimization problem the practitioner can tap into a vast literature on optimization solvers and advanced techniques to speed up the computation. For a reference on the kernel approach in machine learning, see (Cristianini and Shawe-Taylor 2000) or (Schölkopf and Smola 2002). We follow the notation in (Haussler 1999).

[4]The global optimum (hence the learned model) is unique when governed by a positive definite kernel, i.e. $\forall c_1, \ldots, c_N \in \mathbb{R} : \sum_{ij} c_i c_j \mathbf{K}_{ij} > 0$ or equivalently all eigenvalues are strictly positive.

Definition 1.3.1. Positive semi-definite matrix *A symmetric real matrix* $\mathbf{K}$ *is positive semi-definite, iff*

$$\forall c_1, \dots, c_N \in \mathbb{R} : \sum_{ij} c_i c_j \mathbf{K}_{ij} \geqslant 0$$

or equivalently iff all its eigenvalues are nonnegative.

An eigenvalue is a scalar λ such that $\mathbf{K}v = \lambda v$ for some nonzero vector v (the corresponding eigenvector). We will not consider positive semi-definiteness in asymmetric or complex matrices.

Definition 1.3.2. Kernel *Given a set X and a function $K : X \times X \to \mathbb{R}$, we say that K is a* kernel *on $X \times X$ iff it satisfies two conditions:*

- *K is symmetric, i.e. $\forall x, y \in X : K(x, y) = K(y, x)$, and*
- *K is* positive semi-definite*, i.e. for any $N \geqslant 1$ and any $x_1, \dots, x_N \in X$, the matrix defined by $\mathbf{K}_{ij} = K(x_i, x_j)$, is positive semi-definite.*

We will also more concisely say that K is a kernel on X.

If each $x \in X$ can be represented as a feature vector

$$\phi(x) = [\phi_1(x)\phi_2(x) \dots \phi_d(x)]^\top$$

such that K is the ordinary l_2 dot product

$$K(x, y) = \langle \phi(x), \phi(y) \rangle = \sum_{i=1}^{d} \phi_i(x)\phi_i(y)$$

then K is a kernel. This is easy to see: denoting the feature matrix by $\Phi = [\phi(x_1) \cdots \phi(x_N)]$, we can write $\mathbf{K} = \Phi^\top \Phi$, hence $\sum_{ij} c_i c_j \mathbf{K}_{ij} = c^\top \Phi^\top \Phi c = (\Phi c)^\top \Phi c = \sum_i (\Phi c)_i^2 \geqslant 0$.

The converse is also true, under weak assumptions on X and K (Mercer 1909, Courant and Hilbert 1953). That is, a given kernel K can be represented as $K(x, y) = \langle \phi(x), \phi(y) \rangle$ for some choice of feature map $\phi : X \to \mathbb{R}^d$, $d \in \mathbb{N} \cup \{\infty\}$. In particular, it holds for any kernel K over $X \times X$ where X is a countable set. The vector space $\mathbb{R}^d$ induced by ϕ is called the *feature space*. If X is finite, then the square matrix $\Phi^\top \Phi$ that holds all dot products is called the *Gram matrix* or *Gramian*.

There are many operations on kernels that preserve the kernel property. In (Cristianini and Shawe-Taylor 2000) it is explained that the following operations are permitted:

$$K(x,y) = K_1(x,y) + c \qquad c \geqslant 0 \tag{1.3.1}$$

$$K(x,y) = cK_1(x,y) \qquad c \geqslant 0 \tag{1.3.2}$$

$$K(x,y) = K_1(x,y) + K_2(x,y) \tag{1.3.3}$$

$$K(x,y) = K_1(x,y)K_2(x,y) \tag{1.3.4}$$

$$K(x,y) = K_1(\varphi(x), \varphi(y)) \tag{1.3.5}$$

$$K(x,y) = e^{K_1(x,y)} \tag{1.3.6}$$

$$K(x,y) = \frac{K_1(x,y)}{\sqrt{K(x,x)K(y,y)}} \qquad \text{(normalization)} \tag{1.3.7}$$

From Equations 1.3.1-1.3.4 it follows that any polynomial with positive coefficients also preserves the kernel property. We will refer to polynomials or exponentials of previously introduced kernels as composed kernels. There are two more derived kernel types that will be useful in this thesis:

Definition 1.3.3. Haussler zero-extension *If $S \subseteq X$ and K is a kernel on $S \times S$ then K may be zero-extended to be a kernel on $X \times X$ by defining $K(x,y) = 0$ if x or y is not in S.*

Definition 1.3.4. Feature zero-extension *If K is a kernel on $X \times X$ characterized by the feature map ϕ, then K', characterized by ϕ', where*

$$\langle \phi'(x), \phi'(y) \rangle = \sum_{i=1}^{d} \alpha_i \phi_i(x) \phi_i(y) \quad \alpha_i \in \{0,1\}$$

is a zero-extension of K.

From Equation 1.3.5 it is easy to see that any zero-extension of a kernel is a valid kernel.

1.4 Support vector machines

We will briefly sketch the principles of what is likely the most common kernel method in machine learning: the Support Vector Machine (SVM) (Vapnik 1995). A more detailed description can be found in e.g. (Suykens et al. 2002).

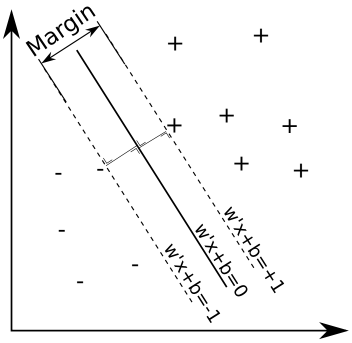


Figure 1.2: The hyperplane of a linear classifier with linearly separable data in a 2D space.

The original setting is that of binary classification. Consider a training set $\{x_k, y_k\}, k = 1 \dots N$ where $x_k \in \mathbb{R}^d$ are the input instances and $y_k \in \{-1, +1\}$ are their respective classes. A linear classifier will be of the form

$$y(x) = \text{sign}(w^\top x + b). \tag{1.4.1}$$

Such a classifier describes a hyperplane $w^\top x + b = 0$ in $\mathbb{R}^d$. Examples on one side of the hyperplane classified as positive, and the others as negative. The Euclidean distance between a point x_k and the hyperplane is $\frac{|w^\top x_k + b|}{\|w\|_2}$, where $\|\cdot\|_2$ denotes the l_2 norm: $\|w\|_2 = \sqrt{w^\top w}$. Unless a training point is on the hyperplane, we can rescale the problem such that $\forall k : |w^\top x_k + b| \geqslant 1$, with the equality holding for the points nearest the hyperplane.

At first, assume that a perfect linear classifier exists. We say that the examples are linearly separable. Figure 1.2 depicts such a situation in a low-dimensional space. Perfect classification implies that

$$\forall k : y_k(w^\top x_k + b) \geqslant 1. \tag{1.4.2}$$

Given linearly separable data, there are infinitely many perfect classifiers. A good and canonical choice is the one which maximizes its *margin*, that is, the one which maximizes the distance between the hyperplane and the nearest positive and negative training examples. This is equivalent to minimizing $\|w\|_2$.

Training the maximum margin classifier involves solving the following optimization problem:

$$\arg\min_{w,b} \frac{1}{2} w^\top w \tag{1.4.3}$$

$$\text{subject to } \forall k \in \{1, \ldots, N\} : y_k(w^\top x_k + b) \geqslant 1 \tag{1.4.4}$$

Using N Lagrange multipliers α_k for the constraints, the primal problem can be reformulated as the equivalent dual problem:

$$\arg\max_{\alpha} -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^\top x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{1.4.5}$$

$$\text{subject to } \forall k \in \{1, \ldots, N\} : \sum_{k=1}^{N} \alpha_k y_k = 0, \quad \alpha_k \geqslant 0 \tag{1.4.6}$$

In the dual problem, the input instances x_k appear only as dot products $x_k^\top x_l$. If the instances x_k are not real vectors (as is the case for molecular structures), we can substitute a suitable kernel for the dot products. The instances are then implicitly mapped to the kernel's feature space.

The dual problem has the form of a convex Quadratic Programming problem (Fletcher 1987). If the problem matrix defined by $K_{kl} = x_k^\top x_l$ is (strictly) positive definite[5], a unique solution is guaranteed. There are no local optima in any case. There exist various efficient algorithms for solving this type of problem.

For most problems, only a limited number of instances lie on the margin. This means that most of the constraints 1.4.4 will be inactive: the left-hand product is strictly larger than one. This, in turn, means that in the dual problem, the corresponding α_k will be zero. The learned model then does not use the corresponding training examples hence we do not need to store them nor compute the corresponding kernel values when evaluating the model.

If the data is not separable, we can add tolerance for misclassifications by introducing slack variables ξ_k in the constraints (Inequality 1.4.4):

$$\arg\min_{w,b,\xi} \frac{1}{2} w^\top w + c \sum_{k=1}^{N} \xi_k \tag{1.4.7}$$

$$\text{subject to } \forall k \in \{1, \ldots, N\} : y_k(w^\top x_k + b) \geqslant 1 - \xi_k, \quad \xi_k \geqslant 0 \tag{1.4.8}$$

[5]Recall that a Gramian is at least positive semi-definite.

The positive constant c governs the bias-variance trade-off: with increasing c, the model mispredicts fewer training examples at the cost of becoming more complex and potentially overfitting the training data.

In the equivalent dual problem, this adds an upper bound to the Lagrange multipliers:

$$\arg\max_{\alpha} -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l x_k^\top x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{1.4.9}$$

$$\text{subject to } \forall k \in \{1, \ldots, N\} : \sum_{k=1}^{N} \alpha_k y_k = 0, \quad 0 \leqslant \alpha_k \leqslant c \tag{1.4.10}$$

Chapter 2

Augmented molecular graphs

Molecular graphs are a compact representation of molecules, but may be too concise to obtain optimal generalization performance from graph-based machine learning algorithms. Over centuries, chemists have learned what are the important functional groups in molecules. In this chapter, we introduce a simple method to incorporate this type of background knowledge in molecular graphs: we insert additional vertices with corresponding edges for each functional group and ring structure identified in the molecule.

In sections 2.4 and 2.5, we investigate in detail the effect of the proposed augmentation method on the predictive performance of existing graph kernel based QSAR models. Experimental evidence on a wide range of ligand-based tasks and datasets shows clear advances of predictive power. When the augmentation technique is used with the recent Pairwise Maximal Common Subgraphs Kernel, we achieve a significant improvement over the current state-of-the-art on the NCI-60 cancer dataset in 28 out of 60 cell lines, with the other 32 cell lines showing no significant difference in accuracy. Finally, on the Bursi mutagenicity dataset, we obtain near-optimal predictions.

This chapter is based on (De Grave and Costa 2010b)[1] and (De Grave and Costa 2010a).

[1]An updated version of (De Grave and Costa 2010b) is reproduced here with permission. Original copyright 2010 American Chemical Society.

2.1 Introduction

Virtual screening is an increasingly important component of the search for novel drug lead compounds (Lengauer et al. 2004). There are two fundamental approaches: target-based[2] and ligand-based. The target is a protein linked to the disease. In target-based virtual screening, the goal is to find compounds which interfere with the normal biological function of the target, usually by reversibly (non-covalently) binding to it. This is achieved by docking: searching a mutual conformation of the target and the potential ligand with sufficiently low binding energy. One can use docking to estimate the affinity only if the 3D structure of the target protein is available in sufficient detail.

In contrast ligand-based screening does not require specific knowledge about the protein structure, but ranks or classifies molecules in a database according to their similarity to known active and inactive molecules. The most critical aspect of accurate ligand classification is how to represent molecules for algorithmic processing (Wale et al. 2008). In this chapter, we describe a novel method for representing molecular graphs. A graph, as we saw in Deold finition 1.2.1, is a mathematical structure, consisting of a set of vertices (also called nodes) and a set of binary relations between vertices, called edges. It is natural to represent a molecule as a graph where atoms are vertices and bonds are edges. In fact, graph theory and chemistry advances are historically tightly linked (Brown 2009). Most small molecules can be losslessly represented by their molecular graph. That is, if care is taken to encode stereogenic centers properly, a synthetic chemist could in principle reproduce the exact molecule from the information contained in an atom-bond graph[3]. Molecular graphs are also a very compact and efficient representation. Common file formats for molecules, such as MDL mol and Sybyl mol2, essentially store an atom-bond graph.

[2]Target-based drug discovery is also known as structure-based, where structure refers to the 3D structure of the target protein.

[3]This holds only for small molecules. Additional information can be embedded in the way a macromolecule is folded, which is not captured in their atom-bond graph alone. Even in some rather small molecules can extra information arise when the conformational space is disconnected by steric hindrance or other forces. The extra information is which partition of the conformational space is occupied by the molecule. Knotanes (Lukin and Vögtle 2005) or molecular knots are a special case of such partitioning. In this thesis, we are concerned primarily with drug-like molecules, which are small and do not usually have a special topology.

2.2 Molecular graph augmentation

2.2.1 Motivation

Even though atom-bond graphs are a representation that is both compact and potentially lossless, they are not necessarily the best representation for a machine learning algorithm to achieve maximal predictive performance. We say that an algorithm has good generalization capabilities when it can predict a property, such as the level of a certain biological activity of unseen instances, after learning from a limited number of observations. When human experts have to make such predictions, they access prior knowledge about the behavior of molecules that has been built up by generations of chemists. The knowledge has been gained by observing numerous phenomena in the past, directly or loosely related to the property under study. The value of the prior knowledge (or background knowledge) can be expressed as a number of "free" extra observations of the property under consideration. To see why, consider the equivalence class notion induced by the donor/acceptor property[4]. Assuming that the activity of a molecule remains unchanged when a donor is present in a specific position, then the set of molecules obtained by swapping functional groups that maintain the presence of the donor atom in that specific position are all equivalent. This equivalence property can be either encoded specifying (in a compact way) the notion of donor/acceptor or by explicitly enumerating all molecular variants that are equivalent.

A computer algorithm, in contrast to the human expert, when presented molecules encoded in their non-redundant atom-bond representation, has access only to the direct observations. It is therefore natural to try to encode the prior knowledge of chemists, either as part of a special-purpose learning algorithm, or as part of the representation of the molecules as they are presented to the algorithm. We will take the latter option. Although the idea of altering the information encoded on the chemical graph is not novel by any means, we will take a novel augmentation approach: we start from the atom-bond graph representation and add background knowledge, more specifically we annotate functional groups and rings as they are described by most chemical textbooks, using extra vertices. In the remainder of the chapter we will use the general term moiety to refer to either a functional group or a small ring.

[4]A hydrogen atom attached to an electronegative atom is a hydrogen bond donor. An electronegative atom is an acceptor. Given suitable geometry, a hydrogen bond may form between a donor and an acceptor. Hydrogen bonds play an important role in the biological activity of molecules.

2.2.2 Adding moiety nodes to an atom-bond graph

It has long been established that chemistry can be explained by underlying laws of physics. Yet only recently it has become clear that the full and exact information about a molecule's behavior is contained in the electron density field (Mezey 2009). However, for practical and computational purposes, the continuous, three-dimensional field must be discretized at some point. Functional groups are essentially an empirically established, discrete set of electron density cloud characteristics that remain fairly constant for a definite group of atoms, independent of their environment (Bader et al. 1994). Functional groups can hence be considered to be an information-dense discretizing approximation of molecular behavior.

DMax (Ando et al. 2006), developed under the lead of Luc Dehaspe, is an Inductive Logic Programming (ILP) (De Raedt 2008c) system with specialized background knowledge to tackle chemical and biological problems. It is related to the ACE relational data mining system (Blockeel et al. 2009), from which it inherited some components, such as the query refinement operator and the Prolog system hipP[5]. The version of DMax for QSAR rule induction is called DMax Chemistry Assistant (DCA). The program finds rules describing (potentially complex) substructures and properties of molecules that are positively or negatively correlated with the measured biological activity. For this purpose, the tool has a sophisticated built-in library to calculate functional groups and rings of a compound (Vandecasteele and Van Craenenbroeck 2002), which it uses as building blocks for more complex rules. The identified moiety instances are stored in a special-purpose relational database. In this chapter, we extract information from this database to construct augmented graphs. All 77 moieties for which nodes are added, are listed in Table 2.1.

DCA defines a hierarchy of moieties to be able to discover activity-correlated rules with the most appropriate specificity. For example, 'any amide group' is more general than 'sulfonamide'. If there is pertinent evidence in the observations, DCA can hypothesize that in a specific location the presence of a sulfonamide is critical (probably in addition to other requirements in nearby locations). However, if the data contains counterexamples that achieve a high level of activity without the sulfone, DCA will allocate more credence to the alternative hypothesis: that any type of amide group is sufficient. In this thesis, we make use only of the most specific moiety definitions. Since we do not explicitly add generalized functional group concepts such as 'any amide group' or 'any ether', we will rely entirely on the machine learning algorithm for generalization. We also exclude composed concepts, such as phenol or urea.

[5]hipP is an acronym for high performance Prolog. It was previously known as ilProlog — Prolog for inductive logic programming.

A possible extension of the method is to use additional nodes to represent more general concepts in the hierarchy, or encode them using extra labels.

It is important to stress that simple subgraph matching is not sufficient to identify moieties, for two reasons:

- The matching may be context-sensitive. For example, a nitrogen is not considered an amine if it is connected to a carbonyl group. Instead, the atoms collectively act as an amide function.
- Some structural variation may be allowed, such as resonance structures or bioisosteres. For example, a hetero-non-aromatic ring may contain any non-carbon atom.

In DCA, functional group identification is implemented by means of logic programming[6]. Some of the functional groups defined in DCA are difficult to express concisely in the popular molecular pattern language SMARTS[7]. This is because DMax's programming language Prolog is Turing-complete[8], whereas SMARTS lacks even variables[9]. The illustrative source code excerpt in Figure 2.1 shows DCA's Prolog code for ether and methoxy. A subexpression for a thiocarbonyl group cannot be predefined in standard SMARTS, so one has to repeat the definition at every location where it is used. The SMARTS query that is equivalent to DCA's ether is `[O([C;!$(*H3);!$(*=O);!$(*=$(S;*!*))]) [C;!$(*H3);!$(*=O);!$(*=$(S;*!*))]]`.

To elucidate the augmentation process, Figure 2.2 shows the composition of the augmented graph of ribavirin, a nucleoside antimetabolite antiviral agent. The basis is the atom-bond graph as defined by the molecule's structural formula, where atoms are vertices labeled with the atom type, and bonds are edges labeled as either single, double, triple, or aromatic. Hydrogen atoms are omitted. We used DCA's aromaticity perception. It verifies Hückel's rule for (systems of) 5- and 6-rings. In the case an O, N, or S atom is included in the ring between single bonds, the ring can also be aromatic. DCA's aromaticity perception is more limited than some other software packages, e.g. it doesn't look at atoms outside of the ring. This limitation influences subsequent perception of moieties, though we did not try to isolate the effect and are hence not sure whether the effect is detrimental to the generalization performance.

[6]For an introduction to logic programming, see e.g. (De Raedt 2008a).

[7]However, one could compose an alternative set of interesting patterns using SMARTS.

[8]Turing-completeness of a programming language means that a Turing machine (Turing 1937) can be implemented in the language. This is equivalent to the notion that any computable function can be programmed in the language. All general purpose programming languages are Turing-complete, and therefore fundamentally equivalent.

[9]Variables are available for the restricted purpose of ring closures. The Daylight SMARTS Toolkit does allow to bind names to subexpressions.

benzene ring	thioamide
pyrrole ring	sulfonamide
furan ring	sulfinamide
thiophene ring	oxime
pyrazole ring	thioxime
imidazole ring	imine
pyridine ring	hydroxylamine
pyridazine ring	thiohydroxylamine
pyrimidine ring	amine
pyrazine ring	n-hydroxyamide
(other) hetero-aromatic ring	n-sulfanylamide
(other) non-hetero-aromatic ring	hydroxyammonium
hetero-non-aromatic ring	sulfanylammonium
non-hetero-non-aromatic ring	ammonium ion
methyl	nitroso
phosphate	thio-S-carboxylic ester
phosphonate	dithiocarboxylic ester
phosphinate	thioether
miscellaneous phosphor	thio-S-carboxylic acid
acylhalide	dithiocarboxylic acid
halide	thiol
carboxylic ester	conjug. base of a thio-S-carboxylic acid
thio-O-carboxylic ester	conjug. base of a dithiocarboxylic acid
methoxy	sulfide
ether	n-hydroxythioamide
carboxylic acid	n-sulfanylthioamide
thio-O-carboxylic acid	sulfoxide
alcohol	sulfinic acid
conjug. base of a carboxylic acid	sulfinic ester
conjug. base of a thio-O-carboxylic acid	conjugated base of a sulfinic acid
oxide	sulfonic acid
ketone	sulfonic ester
aldehyde	conjugated base of a sulfonic acid
diazo	sulfone
azide	metal ion
nitro	counterion
nitrile	(other) heteroatoms
iminium ion	aliphatic chain
amide	

Table 2.1: List of all moiety types introduced during augmentation.

```
functional_group(Name, Path) :-
        atom(Atom1, c),
        sym_bond(Atom1, Atom3, single),
        atom(Atom3, o),
        sym_bond(Atom3, Atom4, single),
        atom(Atom4, c),
        Atom4 > Atom1,
        (
          (
            (AtomT = Atom1; AtomT = Atom4),
            (
              carbonyl_group([AtomT|_])
            ;
              thiocarbonyl_group([AtomT|_])
            )
          ) ->
          ... (omitted)
        ;
          ( (AtomC = Atom1 ; AtomC = Atom4),
              methyl_group([AtomC|RestPath])   ->
              Name = methoxy_group,
              concatenation([Atom3],[AtomC|RestPath],Path)
          ;
              Name = ether,
              Path = [Atom3]
          )
        ).
```

Figure 2.1: Partial DMax source code for methoxygroup and ether. One can see that the matching of the functional groups is context-dependent. E.g., an ether cannot be adjacent to a methyl group.

To obtain the augmented graph, the following steps are performed:

1. Vertices for all moieties defined in the background knowledge are added to the atom-bond graph.

2. We add part-of edges between the moiety vertices and their constituent atoms. Atoms can be part of multiple moieties.

3. The moieties are joined with an edge labeled as: a) *fused* when their constituent atom level subgraphs share one or more vertices; or as b)

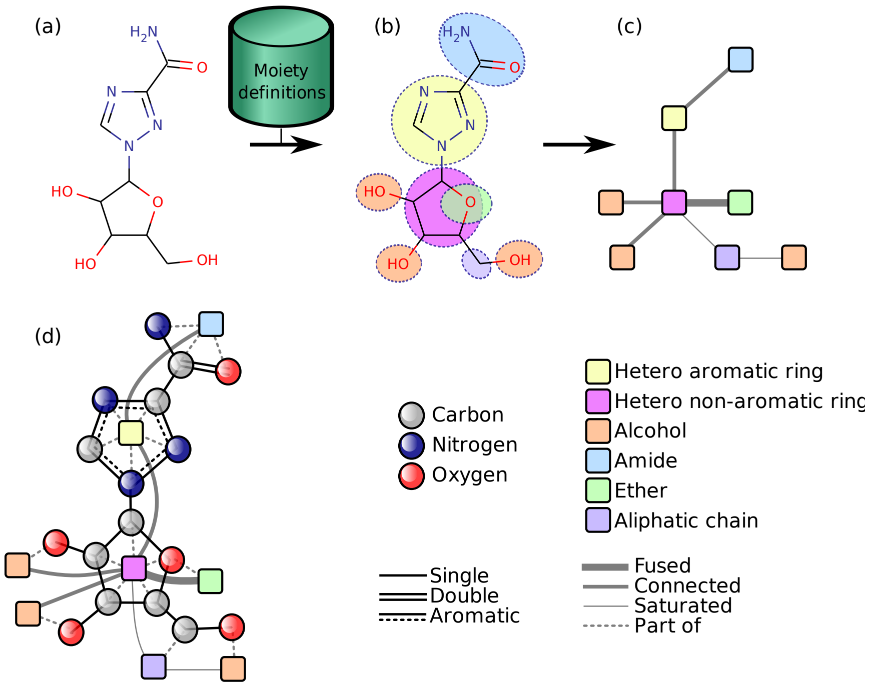


Figure 2.2: Molecular graph augmentation: (a) structure of ribavirin (input), (b) identification of functional groups and rings, (c) the moieties are encoded as extra nodes in the graph, which are added to the original structure to obtain (d) the augmented graph.

connected when their constituent atom level subgraphs do not have any vertex in common, but there exists an edge connecting vertices belonging to the two different moieties.

4. Edges connecting any moiety to an aliphatic chain[10] are not labeled as *connected* but as either c) *saturated* if the chain is saturated[11], or d) *unsaturated* otherwise.

In the ribavirin illustration, the 1,2,4-triazole is not in the list of defined groups and is therefore represented by a vertex labeled as general hetero-aromatic ring.

[10] An aliphatic chain is a maximal group of connected carbon atoms that are not part of a ring or another functional group.

[11] An aliphatic chain is saturated if there are only single bonds between the carbons.

Moieties may share atoms, e.g. in ribavirin the oxygen in the hetero non-aromatic ring also functions as an ether.

As a result of the augmentation, the graph kernels have access to a large subset of the BK-Level-1 background knowledge from (Ando et al. 2006), which primarily consists of the definitions of moieties and their relationships.

Finally, note that the method we present makes use of 2D information only; that is, we do not label stereogenic centers in any way and any model based on our representation is therefore unable to distinguish between stereoisomers.

2.2.3 Related molecular graph representations

Several alternative graph representations of molecules have been described for use in virtual screening. Previous approaches tended to use leaner, more abstract representations rather than an enriched one. A prime example are reduced graphs, first introduced by Gillet et al. for substructure searching and later adapted for similarity searching (Takahashi et al. 1992, Gillet et al. 2003, Barker et al. 2003). The key idea is to omit irrelevant details from the molecule and retain a more abstract graph. A node in the reduced graph may represent multiple atoms in the original graph. Several abstraction types are used, giving rise to different reduced graph types. For example, each ring may be reduced to a single node labeled R, a set of recognized functional groups may indiscriminatively be represented as nodes labeled F, and all other atoms can be mapped to catch-all link nodes if they separate the rings and features, giving rise to a Ring/Feature reduced graph with just three node types. The most detailed level of abstraction for functional groups in (Gillet et al. 2003) are hydrogen donors and acceptors. Reduced graphs allow to find more structurally dissimilar molecules than virtual screening by fingerprinting standard molecular graphs. In the process, unfortunately, most implementations sacrifice some generalization performance. An exception is (Stiefl et al. 2006) which introduced a variant called extended reduced graphs (ErG) with better generalization performance than standard Daylight fingerprints.

The most similar approach to the method proposed here may be the one by (Takahashi et al. 1992). There, the authors compute a graph of specific functional groups, unlike later work where more abstract concepts have been used. The vertices are labeled with topological distances (multiple if there are different paths).

The representation of molecules by feature trees was first proposed in (Rarey and Dixon 1998). A molecule graph is converted to a tree by iteratively collapsing its minimal-length cycles into single nodes. Each node in the tree is associated with a vector of chemical features, such as approximated Van Der Waals volume and hydrogen donorship. The features for larger subtrees can be computed from the features of its parts, e.g. by summation. A similarity function must be provided for each feature. The direct similarity of two subtrees is computed as the weighted sum of the similarity of their features. For comparing two molecules, the algorithm tries to find a coordinated way of splitting the molecules in subtrees such that the highest aggregate direct similarity of matched subtrees is obtained. The tree representation is a theoretically attractive middle ground in between graphs, where most operators are computationally expensive, and vectors, which lack expressiveness. As opposed to the feature trees algorithm, the approach taken in this thesis separates the representational concern from the similarity computation. This allows us to delegate the computation of the similarity to a graph kernel, which is positive semi-definite, a property so far not attributed to feature tree similarity scores. A kernel, unlike a non-kernel similarity score, allows the direct use of kernel-based machine learning models, such as support vector machines.

2.3 Graph kernels

In this chapter, we investigate the effect of graph augmentation on the QSAR modeling performance of kernel-based classifiers. Kernel methods have proved to achieve excellent generalization performance in many machine learning tasks. Recall from Section 1.3 that a kernel is essentially a similarity measure that possesses the mathematical properties of symmetry and positive semi-definiteness. It is associated with a feature map. The kernel function values correspond to the scalar product of the image of two instances translated to the feature space. Vice versa, the ability to describe a similarity measure as a scalar product in some vector space guarantees that the similarity is a kernel. It is often convenient to describe a kernel in terms of its feature map.

Here, we are concerned with small molecule classification, therefore we look into kernel methods for graphs, for which an increasingly large literature exists (see (Gärtner 2003) for references). In this chapter, we compare three different types of graph kernels: the Equal Length Shortest-Path Kernel, the Weighted Decomposition Kernel, and the Pairwise Maximal Common Subgraphs Kernel. The choice is motivated by the desire to sample diverse approaches within the graph kernel techniques: the Equal Length Shortest-Path Kernel considers long distance interactions between pairs of vertices in a graph, the Weighted Decomposition Kernel considers the local information in the neighborhood of the vertices in a graph, while, finally, the Pairwise Maximal Common Subgraphs Kernel considers the occurrence of shared large subgraphs.

2.3.1 Equal Length Shortest-Path Kernel

An efficient yet effective family of graph kernels was proposed in (Borgwardt and Kriegel 2005). We will concentrate on the variant with the best accuracy-speed trade-off, the equal length shortest-path kernel (ELSPK). In this thesis, we use our own implementation of the (zero-extended, see below) ELSPK, which we will call pairwise distance kernel (PDK) for reasons that will become clear in Chapter 3.

The idea is to compute the similarity between two graphs by comparing all the respective pairs of vertices annotated with their topological distances. This is achieved by

1. calculating the shortest path distance between all pairs of vertices using Floyd-Warshall's algorithm (Floyd 1962, Warshall 1962); and subsequently
2. computing an all-pairs-shortest-paths kernel on edge walks of length 1 on an appropriately modified graph.

Formally, a graph G is transformed into a graph S such that there exists an edge between two nodes in S if they are connected by a path in G (i.e. S is the complete graph of the vertex set of G when G is connected). Every edge in S is labeled by the shortest distance between these two nodes, and we denote it with the term *"distance-edge"*. Given the vertex set V_S and edge set E_S, the ELSPK is defined as:

$$K(S, S') = \sum_{e \in E_S} \sum_{e' \in E'_S} k^{(1)}(e, e') \tag{2.3.1}$$

where $k^{(1)}$ is a positive semi-definite kernel on edge walks of length 1.

The choice to require equal lengths means that $k^{(1)}$ is the exact matching kernel over edges, where two edges match if they have the same label and if the labels of their vertices also match:

$$k^{(1)}(e, e') = \delta(e, e') = \begin{cases} 1 & \text{if } \lambda(v) = \lambda(v') \text{ and } \lambda(u) = \lambda(u') \text{ and } \lambda(e) = \lambda(e') \\ 0 & \text{otherwise} \end{cases} \tag{2.3.2}$$

where $e = uv$, $\lambda(v)$ is the vertex label, and $\lambda(e)$ is the edge label, i.e. in this case the topological distance between v and u in G.

To obtain PDK, we specialize $k^{(1)}$ to its zero-extension parameterized by a maximum distance d, that is, we consider:

$$k_d^{(1)}(e, e') = \begin{cases} k^{(1)}(e, e') & \text{if } \lambda(e) = \lambda(e') \leq d \\ 0 & \text{otherwise} \end{cases} \tag{2.3.3}$$

Hence, PDK considers all pairs of vertices up to a maximum distance d and counts how many exact matches there are between the distance-edge sets representing the two original graphs. In Figure 2.3 we give a graphical representation of the distance-edge set induced considering a given single vertex.

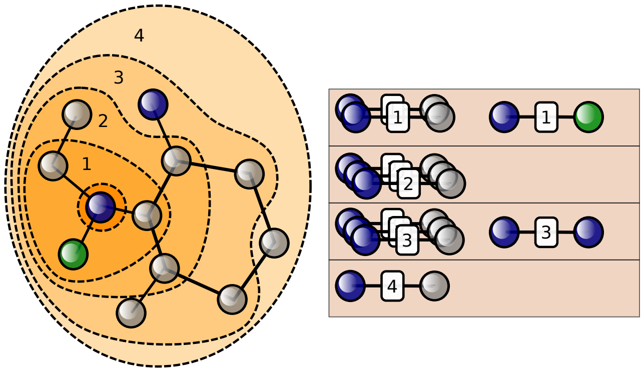


Figure 2.3: The set of distance-edges induced by a single vertex for the ELSPK (or PDK). On the left, the original graph G is depicted. The table on the right shows only the descriptors induced by the vertex of interest, the highlighted nitrogen. The kernel will complete the table for all other vertices. Two molecules are then compared by counting the elements in the intersection of their respective tables.

2.3.2 Decomposition kernels

We first introduce the general class of decomposition kernels (or convolution kernels) to facilitate the explanation of the Weighted Decomposition Kernel.

Let $x \in X$ be a *composite structure*. We define $x_1, \ldots, x_D$ to be the parts of x. The set of parts needs not be a partition of the composite structure, i.e. the parts may "overlap".

Each part x_d is in a set X_d for $d = 1, \ldots, D$, with $D \in \mathbb{N}^*$, and each X_d is a countable set. Let R be the "parts-of" relation, defined on the set $X_1 \times \ldots \times X_D \times X$, such that $R(x_1, \ldots, x_D, x)$ is true iff $x_1, \ldots, x_D$ are the parts of x. We denote with $R^{-1}(x)$ the inverse relation that yields the parts of x, that is:

$$R^{-1}(x) = \{x_1, \ldots, x_D : R(x_1, \ldots, x_D, x)\} \tag{2.3.4}$$

In (Haussler 1999) it is demonstrated that, if there exist a kernel K_d over $X_d \times X_d$ for each $d = 1, \ldots, D$, and if two instances $x, y \in X$ can be decomposed in $x_1, \ldots, x_d$ and $y_1, \ldots, y_d$, then the following generalized convolution:

$$K(x,y) = \sum_{\substack{x_1,\ldots,x_d \in R^{-1}(x) \\ y_1,\ldots,y_d \in R^{-1}(y)}} \prod_{d=1}^{D} K_d(x_d, y_d) \tag{2.3.5}$$

is a valid kernel called a *convolution* or *decomposition* kernel[12]. A decomposition kernel is thus a sum (over all possible ways to decompose a structured instance) of the product of valid kernels over the parts of the instance.

2.3.3 Weighted Decomposition Kernel

The Weighted Decomposition Kernel (WDK) is a specialization of a decomposition kernel, introduced in (Menchetti et al. 2005). The idea is to compare not just individual vertices, but a larger *context* associated with each vertex. More precisely, each vertex v in a graph G is characterized by a context, which is a region of topological nearby elements, not farther from v than a predefined maximum distance l. The similarity between two graphs is then computed in terms of the similarity of the set of their vertices V_x weighted by the similarity of the their respective contexts.

Formally:

$$K(G,G') = \sum_{v \in V_G} \sum_{v' \in V_{G'}} \delta(v,v') \cdot k^{(l)}(v,v') \tag{2.3.6}$$

where the similarity between two vertices v and v' is computed by the exact matching kernel:

$$\delta(v,v') = \begin{cases} 1 & \text{if } \lambda(v) = \lambda(v') \\ 0 & \text{otherwise} \end{cases} \tag{2.3.7}$$

where $\lambda(v)$ is again the vertex label.

The context kernel $k^{(l)}$ is a set kernel computed over the edges in the neighborhood subgraphs:

$$k^{(l)}(v,v') = \sum_{e \in E_{\Omega_l^v}} \sum_{e' \in E_{\Omega_l^{v'}}} \delta(e,e') \tag{2.3.8}$$

[12]To be precise, the valid kernel is the zero-extension of K to $X \times X$ since $R^{-1}(x)$ is not guaranteed to yield a non-empty set for all $x \in X$.

where $\delta(e, e')$ was defined in Equation 2.3.2 and the neighborhood subgraph Ω_l^v in Definition 1.2.13.

The similarity of two node contexts is thus defined as the number of exact matches between the edges present in the contexts of v and v'. In Figure 2.4 we give a graphical representation for the context edge set of two vertices.

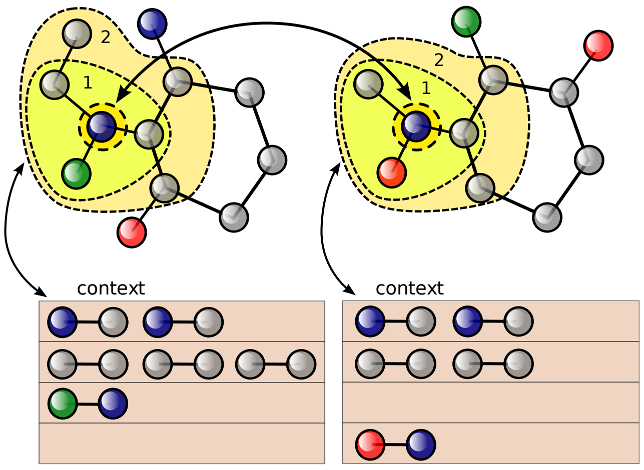


Figure 2.4: Context edge set for two vertices according to WDK with context radius $l = 2$. The two highlighted nitrogens can contribute to the similarity of the molecules because their atom type matches. How much they contribute, depends on the context they occur in: it is the number of exact matching edges in their respective contexts, which is 4.

2.3.4 Pairwise Maximal Common Subgraphs Kernel

The Pairwise Maximal Common Subgraphs Kernel (PMCSK) (Schietgat et al. 2009, Schietgat et al. 2011, Schietgat 2010) is a kernel built over structural keys that is computed in two steps: at first a set of relevant subgraphs is extracted from all possible pairs of instances; the subgraphs are then used to provide a bit vector encoding for a graph in the following way: the bit at position i is asserted if the i-th subgraph is present in the graph; finally the similarity between two graphs is computed via a Jaccard similarity coefficient (Jaccard 1901).

The Jaccard similarity coefficient, also known as Jaccard index and as Tanimoto coefficient (which generalizes the similarity index to real-valued features

(Tanimoto 1957)), is considered a state-of-the-art similarity score for the classification of small molecules based on fingerprints (Willett 2006). The similarity score is computed by counting the number of common elements (i.e. the set-intersection) between the two instances as a fraction of the total number of elements that occur in both instances (i.e. the set-union). Formally, if an instance x has $|x|$ bits asserted, x' has $|x'|$ bits asserted and they share $|x \wedge x'|$ asserted bits, then the Tanimoto similarity score is a real number in the interval $[0, 1]$ computed as:

$$\frac{|x \wedge x'|}{|x| + |x'| - |x \wedge x'|}.$$

This similarity score is clearly symmetric. In (Gower 1971, page 868) it is proved[13] that it is also positive semi-definite, so it satisfies the Mercer conditions for a Mercer kernel. It is hence appropriate to use the terms 'Tanimoto kernel' and 'Pairwise Maximal Common Subgraphs Kernel'.

The novelty of the PMCSK lies in the definition of the relevant/interesting subgraph: a subgraph is relevant if it is the maximal common subgraph between two instances belonging to the dataset. This criterion differs from the usual structural keys approaches in that it does not use a pre-defined dictionary of fragments, nor does it consider the set of *all* fragments up to a predefined maximum size.

Computing the maximal common subgraph in the general case is an NP-hard problem. Fortunately, there exists a polynomial-time algorithm if one considers only outerplanar graphs in combination with the block-and-bridge-preserving (BBP) subgraph isomorphism (Schietgat et al. 2008).

Definition 2.3.1. Planar graph *A graph is* planar *if it has a planar embedding, that is, it can be drawn in the plane in such a way that no two edges intersect except at a vertex in common. The regions formed by the edges in a planar embedding are called* faces*. There is one unbounded face, which is called the outer face.*

Definition 2.3.2. Outerplanar graph *An* outerplanar *graph is a planar graph that can be embedded in the plane in such a way that all of its vertices lie on the boundary of the outer face.*

Intuitively, a graph is outerplanar when it can be embedded in the plane in such a way that all of their vertices lie on the outside of the graph. In Figure 2.5 a) we give an example of a non-outerplanar molecule: here the graph cannot be drawn on the plane in such a way that the highlighted vertex is reachable from

[13]The Jaccard index corresponds to the dichotomous variates case of Gower's similarity coefficient.

the outside of the graph. Since the vast majority (≈90% of the molecules for each of the four datasets used in this chapter) of small molecule graphs are outerplanar, this restriction does not represent a severe limitation in practice. This was first observed by (Horváth et al. 2006). While non-outerplanar graphs do not contribute to the identification of the MCSs, the presence of the MCSs extracted from pairs of outerplanar graphs is still used to build their vector encodings.

Finally, the BBP subgraph isomorphism is a special case of the general subgraph isomorphism. In BBP isomorphism we distinguish special subgraph configurations called *blocks* and *bridges*. A block is a maximal set of edges such that every pair (of these edges) belongs to a common cycle. In chemical terms, a block is a ring system. A bridge is an edge such that the number of connected components increases when the edge is removed. In chemical terms, bridges are bonds that are not in a ring. The BBP isomorphism prescribes that only bridges of a graph G can be mapped to bridges of the other graph G' and edges of blocks of G can be mapped only to edges of blocks of G'.

In Figure 2.5 b) and c) we give an example of the consequences on the identification of a maximal common subgraph under the general notion of subgraph isomorphism and under the BBP notion.

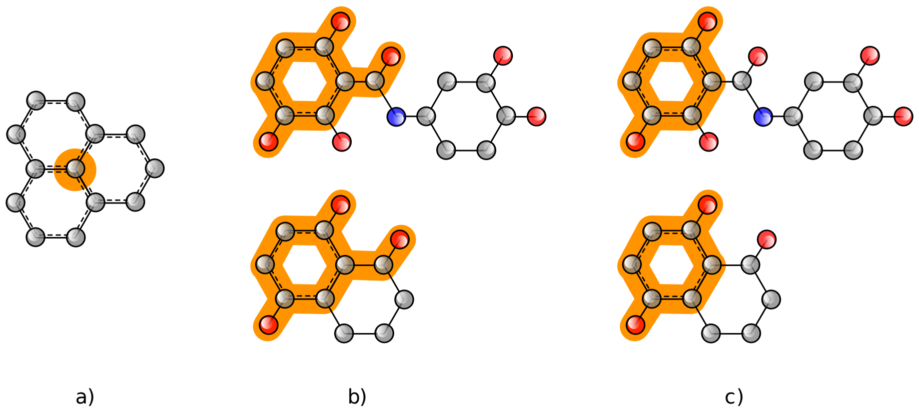


Figure 2.5: a) Example of a non-outerplanar graph; b) a maximal common subgraph under general subgraph isomorphism; c) a maximal common subgraph under BBP subgraph isomorphism

An additional point in favor of the PMCSK is that the block-and-bridge-preserving subgraph isomorphism seems to produce better quality subgraphs for several chemoinformatics tasks when compared with the general subgraph isomorphism; that is, it has been experimentally observed that the induced predictive models exhibit better performances (Schietgat et al. 2009) when the subgraphs are extracted under the BBP isomorphism.

2.4 Experimental setup

2.4.1 Datasets

To determine the utility of the augmentation method, we selected four datasets from three different domains: oncology, virology, and toxicology (in vivo and in vitro).

NCI-60

The Developmental Therapeutics Program (DTP) at the U.S. National Cancer Institute (NCI) has checked a large number of compounds for evidence of the ability to inhibit the growth of human tumor cell lines[14]. The discretised and roughly balanced subset used by (Swamidass et al. 2005) has become a popular binary classification benchmark for QSAR algorithm research, often referred to as NCI-60, NCI60, or just NCI. The benchmark dataset contains growth inhibition measurements on 60 cell lines, even though DTP had previously evicted one of those cell lines because it was essentially a replicate of another (Nishizuka et al. 2003). Each cell line has inhibition data on about 3500 compounds. There are 3910 distinct compounds in the set in total.

HIV

The DTP also runs an AIDS antiviral screening, which has checked a large number of compounds for evidence of protection against HIV-1. The October 1999 release of the database[15] contains the structures of 42687 molecules. Each of the compounds was tested twice, and 422 were confirmed to be active (CA), 1081 are moderately active (CM), and 41184 are inactive (CI). Sometimes, Kramer's subset of 41768 molecules (Kramer et al. 2001) is used to benchmark machine learning algorithms. The full set has also been used before, e.g. in (Ceroni et al. 2007) There are three binary classification tasks commonly considered for this dataset: distinguishing between CA and CM, between CA+CM and CI, and between CA and CI.

[14]`http://dtp.nci.nih.gov/docs/cancer/cancer_data.html`
[15]`http://dtp.nci.nih.gov/docs/aids/aids_data.html`

Binary class	Original class	Simplified description
positive	CE	Clear Evidence of carcinogenic activity
	P	Positive
	SE	Some Evidence of carcinogenic activity
negative	NE	No Evidence of carcinogenic activity
	N	Negative
(omitted)	IS	Inadequately Studied
	EE	Equivocal Evidence
	E	Equivocal

Table 2.2: Classes in the Predictive Toxicology Challenge. Source: `http://www.predictive-toxicology.org/data/ntp/ntp_results.txt`

PTC

The 2000-2001 Predictive Toxicology Challenge (PTC) (Toivonen et al. 2003) was devised to stimulate the development of machine learning techniques for predictive toxicology models. The data originates from the US National Toxicology Program (NTP). The training and test sets have a different class distribution and a different prevailing mode of action (Benigni and Giuliani 2003), therefore we only use the (corrected) training set, which contains 417 molecules. The aim is to predict the carcinogenicity of the compounds in different rodents, in particular male mice (MM), female mice (FM), male rats (MR), and female rats (FR). Because the PTC dataset is small compared to the three other datasets, we will report the performance of algorithms only as averages over the four tasks. We use the binary classification version of PTC, with the classes CE, P, and SE translated to positive, and the classes N and NE translated to negative (cfr. Table 2.2).

Bursi

Kazius *et al.* (Kazius et al. 2005) have constructed a dataset of 4337 molecular structures with corresponding Ames data[16]. Ames is a short-term in vitro assay designed to detect genetic damage caused by chemicals and has become the standard test to determine mutagenicity. The distribution is 2401 mutagens and 1936 nonmutagens.

[16]`http://www.cheminformatics.org/datasets/bursi/`

2.4.2 Evaluation measures

We will evaluate the generalization performance of the kernels and the augmentation method by the area under the receiver operating characteristic (AUROC) (Gribskov and Robinson 1996). The ROC is the plot of the true positive rate (recall) versus the false positive rate (false alarm rate). An AUROC score of 100% indicates perfect separation of positives from negatives, whereas a score of 0% indicates that all negatives were selected before the first positive. An algorithm that predicts a random order has an expected AUROC of 50%. This holds for any class distribution, because the ROC is based on rates rather than absolute counts.

We also report the ROC_{50} score (Gribskov and Robinson 1996), which is the area under the ROC up to the first 50 false positives[17]. Figure 2.6 illustrates both AUROC and ROC_{50}. The ROC_{50} provides similar insights as *lift* graphs and gives an idea of how reliable the predictions that the method considers most trustworthy effectively are. A ROC_{50} score of 100% again indicates perfect separation of positives from negatives, whereas a score of 0% indicates that none of the top 50 molecules selected by the algorithm were true positives. The expected ROC_{50} of a random prediction algorithm depends on the number of negatives N in the test set:

$$\mathbb{E}_{randompred}(ROC_{50}) = \begin{cases} 25/N & \text{if } N \geqslant 50 \\ 50\% & \text{otherwise} \end{cases} \tag{2.4.1}$$

2.4.3 Experimental goal and setup

We are now well equipped to perform experiments to answer two questions about the proposed methods:

Q1 Does the augmentation of molecular atom-bond graphs with moieties improve the predictive performance of graph kernels in a support vector machine?

Q2 How do augmented graph kernels compare to the current QSAR state-of-the-art?

[17]In order to avoid the size of the dataset having an excessive impact on the score, the horizontal axis is rescaled such that a value of 1.0 on the axis corresponds to the 50th false positive.

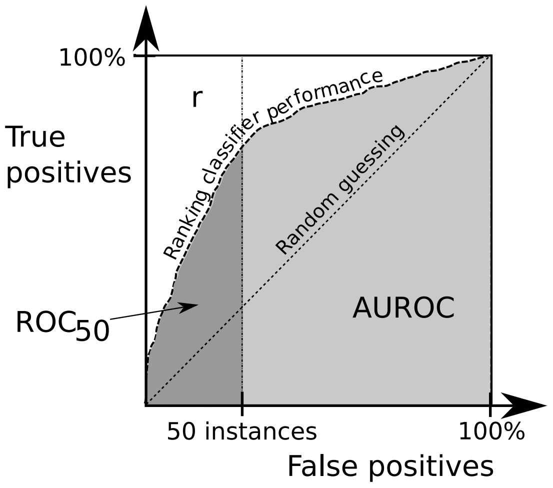


Figure 2.6: ROC curve. The AUROC of the classifier is the entire shaded area, given that the graph is plotted in the unit square. The ROC_{50} score is the dark shaded area divided by the area of rectangle r.

We tested the predictive performance of the SVM-Light (Joachims 1999) support vector machine implementation by 10-fold cross-validation. The folds were always stratified[18] and identical for all methods.

SVM-Light was run with all parameters at their default value except for the cost factor (-j), which was set to the prevalence ratio of negative to positive examples as suggested by (Morik et al. 1999). Each of the kernels was normalized (Equation 1.3.7).

In the following we report method-specific observations.

PDK

The maximum distance parameter d has been optimized by inner cross-validation for each training fold of the Bursi dataset. The range considered was 4, 6, 8, . . . 26. For the other datasets, we used the value that was selected most frequently in the Bursi dataset: 14 for atom-bond graphs and 24 for augmented graphs.

[18]In stratified cross-validation, the number of examples of a given class is the same (plus or minus one) in each test set.

WDK

We used a context radius l of 4 for the WDK, motivated by the results on the HIV dataset by (Menchetti et al. 2005).

In this chapter, in order to have a clearer comparison, we employ a simpler setup than in (Menchetti et al. 2005). In particular we 1) do not compound the kernel with a Gaussian kernel, 2) we do not use the information on the partial charge over single atoms, and 3) we do not use the information about the context complement (that is we do not weigh the similarity of two nodes by considering the edge set similarity of the edges that are *not* part of the vertex context). In this way the analysis of the advantages and disadvantages of different methods can be compared on a clearer and fairer basis since the only information used stems from the atom and bond types.

PMCSK

Unfortunately the molecular graph augmentation procedure as detailed above cannot be directly used with the PMCSK since the presence of the *part-of* edges in the augmented graphs make them non-outerplanar. To circumvent this difficulty, we operated as follows: the MCS descriptor generation method was run separately on the atom-bond graphs and the moiety graphs, both of which are outerplanar on their own for the vast majority of drug-like molecules. More specifically we observe that 90-92% of the atom-bond graphs are outerplanar in all datasets considered and that the fraction of the moiety graphs that are outerplanar ranges from 73% in the HIV dataset to 91% in the PTC dataset.

Finally, the bit-vector representations for the atom-bond graphs and the moiety graphs are concatenated in order to obtain the overall joint representation. We will use the notation PMCSK(G'_{aug}) for this approach.

To reduce the runtime for the HIV dataset, only the relatively small set of "positive" molecules (either CA or CA+CM) were used to derive maximal common subgraphs.

2.5 Results and discussion

2.5.1 Q1: Does the augmentation of molecular atom-bond graphs with moieties improve the predictive performance of graph kernels in a support vector machine?

Table 2.3 shows the average per-fold cross-validated AUROC and ROC_{50} scores for the three described kernels on all four datasets, both for atom-bond graphs and for augmented molecular graphs. The standard deviation over the ten folds is also shown. Note that the number of folds influences the height of the reported ROC_{50}, since a higher number of folds means smaller test sets, while the cutoff number of false positives stays at 50. Indeed, for PTC, the ROC_{50} for each of the 10 folds is equal to the AUROC due to the small number of instances.

The answer to **Q**1 is clear from Table 2.3: augmentation was substantially beneficial with regard to generalization capacity. All tested tasks and all kernels benefit from augmentation. When computing the *relative error reduction*

$$\mathrm{RER} = \frac{e - e'}{e}$$

where e is the error of the original method and e' is the error of the novel method[19], we observe an average reduction of 20% for the WDK and PDK (with a remarkable 45% error reduction when the PDK is used on the Bursi dataset) while the PMCSK presents a more modest 5% average error reduction. Note that diminishing marginal returns are to be expected as performance increases. The ROC_{50} results confirm that the augmentation is also effective in increasing the performance for what are considered the most trustworthy predictions by the WDK and PDK methods with an average RER of 17% (4% for the PMCSK).

Note that the WDK results reported in Table 2.3 are worse than those obtained in (Ceroni et al. 2007) since we use the basic WDK and did not implement all refinements of (Menchetti et al. 2005).

Due to space and time constraints, there obviously remain a large number of different graph kernel approaches, such as (Gärtner et al. 2007, Riesen and Bunke 2009, Rupp et al. 2007), for which we do not obtain direct experimental evidence whether the augmentation procedure presented in this chapter leads to significant predictive performance increase. As a general remark, we note

[19]Note that we have here abused the notation and consider $e = 1-\mathrm{AUROC}$ rather than $e = 1-\mathrm{accuracy}$

Table 2.3: Predictive performance of the three kernels on unaugmented and augmented molecular graphs: average and standard deviation of the AUROC and ROC_{50} scores over 10 folds. NCI-60 and PTC numbers are averages over 60 and 4 tasks, respectively. For comparison, the expected ROC_{50} for random predictions is also shown.

	NCI-60	HIV	HIV	HIV	PTC	Bursi
	(avg.)	CA vs. CM	CACM vs. CI	CA vs. CI	(avg.)	
AUROC (%)						
PDK(G)	70.2 ± 2.5	77.4 ± 2.5	76.0 ± 2.2	90.1 ± 4.5	59.8 ± 8.5	76.5 ± 1.2
PDK(G_{aug})	74.7 ± 2.5	80.1 ± 5.7	81.3 ± 2.3	92.8 ± 2.5	64.7 ± 9.5	87.0 ± 1.5
WDK(G)	73.3 ± 2.5	77.7 ± 5.8	80.5 ± 2.3	92.5 ± 3.6	62.3 ± 10	83.6 ± 1.5
WDK(G_{aug})	77.9 ± 2.4	82.0 ± 4.8	83.4 ± 2.2	94.5 ± 2.9	66.1 ± 9.7	87.6 ± 1.1
PMCSK(G)	79.6 ± 2.2	82.6 ± 6.2	81.8 ± 2.2	93.0 ± 3.7	64.5 ± 8.8	90.5 ± 1.3
PMCSK(G'_{aug})	80.3 ± 2.2	82.8 ± 6.2	83.2 ± 2.1	93.4 ± 3.4	65.6 ± 8.8	91.5 ± 1.1
ROC_{50} (%)						
$\mathbb{E}(ROC_{50})$	15.7	23.1	0.6	0.6	50.0	12.9
PDK(G)	37.3 ± 4.5	60.5 ± 6.0	5.7 ± 2.2	23.9 ± 4.6	59.8 ± 8.5	45.3 ± 2.9
PDK(G_{aug})	42.0 ± 4.8	64.4 ± 7.9	13.5 ± 3.5	39.1 ± 6.5	64.7 ± 9.5	63.8 ± 3.7
WDK(G)	40.2 ± 4.4	60.5 ± 9.1	10.5 ± 2.1	29.7 ± 6.6	62.3 ± 10	58.6 ± 3.5
WDK(G_{aug})	47.1 ± 4.7	68.8 ± 7.6	19.9 ± 2.6	59.2 ± 6.0	66.1 ± 9.7	67.0 ± 3.9
PMCSK(G)	51.7 ± 4.5	71.5 ± 9.4	34.2 ± 3.3	66.0 ± 6.3	64.5 ± 8.8	72.4 ± 3.7
PMCSK(G'_{aug})	52.8 ± 4.5	72.2 ± 9.6	35.5 ± 3.1	67.7 ± 6.9	65.6 ± 8.8	74.8 ± 3.5

Table 2.4: Descriptive statistics of unaugmented and augmented molecular graphs for the Bursi dataset. With avg. degree(V_G) we mean $\sum_{v \in V_G} \frac{\text{degree}(v)}{|V_G|}$.

	Mean	Min	Quartile 1	Quartile 2	Quartile 3	Max
avg. degree(V_G)	2.12	0	2	2	3	4
avg. degree($V_{G_{aug}}$)	3.78	0	3	3	4	81
$\|V_G\|$	16.9	2	11	16	21	214
$\|V_{G_{aug}}\|$	23.08	4	15	21	28	294
$\|E_G\|$	17.88	1	11	17	23	217
$\|E_{G_{aug}}\|$	43.61	4	25	39	56	542

that the augmentation affects several key characteristics of the input graphs, for example on the Bursi dataset we observe the following changes: the vertex and edge label alphabet is increased from 13 to 69 and from 4 to 9 respectively; the vertex degree distribution and the vertex and edge count distribution changes as shown in Table 2.4. For some graph kernels these differences (increased average label alphabet and degree size, and number of vertices and edges) can lead to a significant increase in the expected runtime, a negative aspect that has to be weighted against the expected performance increase. For example, both the method proposed in (Riesen and Bunke 2009) and the one proposed in (Rupp et al. 2007), make use of the Kuhn-Munkres assignment algorithm (Munkres 1957) that has a $O(|V|^3)$ complexity. The method from (Horváth et al. 2004) instead counts the number of cycles in a graph and can suffer from the many cycles introduced by the *part-of* edges added by the augmentation procedure. Here, as in the PMCSK case, a possible workaround would be to eliminate such edges and consider the moiety graph as a disconnected component w.r.t. the original chemical graph.

For the three types of kernels that we have selected, we observe only a modest runtime overhead: 1.5 times for PDK and WDK and negligible for PMCSK. This latter result can be explained by the small size of the moiety graphs; the number of vertices (edges) is approximately one third of the number of vertices (edges) in the standard molecular graphs. As a consequence, the additional runtime spent by the PMCSK algorithm on the moiety graphs is two orders of magnitude lower than the time spent for the standard molecular graphs.

In Table 2.5 we report the runtime (in seconds) required for the augmentation pre-processing step compared to the actual kernel computation. Obviously, the augmentation step is of linear complexity in the number of molecules, while the Gram matrix computation is quadratic. The throughput on the large HIV dataset of the latter, dominant step is about 430,000 molecule-molecule

Table 2.5: Runtime cost of graph augmentation: CPU time in seconds to calculate the Gram matrix for each kernel, without and with augmentation. The time required for augmentation itself is indicated separately.
(*) For the PMCSK on the HIV data, the subgraphs were derived only from the 1503 confirmed active or moderately active molecules.

	NCI-60	HIV	PTC	Bursi
Number of molecules	3910	42687	417	4337
Augmentation time	$3.5 \cdot 10^2$	$3.4 \cdot 10^3$	$3.4 \cdot 10^1$	$1.2 \cdot 10^2$
PDK(G)	$4.2 \cdot 10^1$	$3.9 \cdot 10^3$	$1.0 \cdot 10^0$	$3.6 \cdot 10^1$
PDK(G_{aug})	$7.7 \cdot 10^1$	$4.2 \cdot 10^3$	$2.0 \cdot 10^0$	$5.7 \cdot 10^1$
WDK(G)	$1.8 \cdot 10^3$	$1.6 \cdot 10^5$	$8.0 \cdot 10^0$	$1.1 \cdot 10^3$
WDK(G_{aug})	$2.3 \cdot 10^3$	$2.3 \cdot 10^5$	$1.4 \cdot 10^1$	$1.5 \cdot 10^3$
PMCSK(G)	$2.8 \cdot 10^5$	$3.3 \cdot 10^4$ (*)	$6.2 \cdot 10^2$	$3.5 \cdot 10^5$
PMCSK(G'_{aug})	$2.8 \cdot 10^5$	$3.3 \cdot 10^4$ (*)	$6.3 \cdot 10^2$	$3.5 \cdot 10^5$

comparisons per second for PDK, 8,000 for WDK, and just 70 for PMCSK. The programs were executed on an Intel Core2 Quad Q9550 CPU (2.83GHz), except for the HIV dataset which was run on an Intel Xeon E5420 CPU (2.5GHz) due to 64-bit support of the operating system. The programs are all essentially single-threaded. The table shows net consumed CPU time, except for the augmentation process where we were only able to measure wall clock time. The time for the PDK is for our own implementation; its performance characteristics may bear little resemblance to Borgwardt *et al.*'s original ELSPK. In (Borgwardt 2007) a runtime for ELSPK on PTC is reported which is an order of magnitude higher than PDK's runtime.

2.5.2 Q2: How do augmented graph kernels compare to the current QSAR state-of-the-art?

NCI-60 The best published kernel for the NCI-60 datasets before our own results in (De Grave and Costa 2010b) and (Costa and De Grave 2010) was the Graph Fragments (GF) kernel described in (Wale et al. 2008). For comparison with the state-of-the-art, we used the GF kernel on NCI-60 on the same cross-validation folds as for the other kernels. Graph Fragments, as it is implemented by its authors, is not a general graph kernel, but is highly specialized for molecular atom-bond graphs. For example, one of the built-in constraints is that nodes cannot have a degree larger than 5. This prevented us from using the kernel on the augmented graphs, or even the separate moiety graphs as for the

PMCSK. Furthermore, we did not implement the length-differentiated min-max kernel, but rather used the descriptors produced by the AFGen program in the same experimental settings as for all other kernels.

According to the binomial sign test at the 5% level, PDK performed worse than GF on all cell lines, except one where there was no significant difference. WDK performs worse than GF in 25 cell lines, while for the remaining 35 cell lines the null hypothesis of equal performance could not be rejected. PMCSK, however, performed significantly better than GF in 28 cell lines and worse in none.

HIV WDK (Menchetti et al. 2005) is the method with the highest reported AUROC (84.2%) for the CA vs. CM task in HIV. In the recent paper (Swamidass et al. 2009) an AUROC of 84.5% was reported on the CA+CM vs. CI task. Wale *et al.* report an AUROC of 95.0% for the CA vs. CI task, using Acyclic Fragments.

PTC Wale *et al.* also report an AUROC of 71.1% for PTC using Path Fragments and 71.0% using GF.

Bursi To our knowledge, the highest AUROC on the Bursi dataset was obtained by Saigo *et al.*, who report a best AUROC of 88.9% using gBoost (Saigo et al. 2009), and an accuracy of 82.5%. The training accuracy of 83% in (Kazius et al. 2005) was achieved with a manually constructed model, using all data without cross-validation. However, Kazius *et al.* do use a smaller, separate test set, where the model counterintuitively achieves a slightly higher accuracy of 85%. They report that the average interlaboratory reproducibility error is 15%, which provides an approximate upper bound on the achievable accuracy. The accuracy obtained with PMCSK(G'_{aug}) was 85%, equal to the estimated upper bound.

Finally, the answer to **Q**2 is that the PMCSK operating on the augmented molecular graphs exhibits a predictive performance that is competitive with state-of-the-art results, in particular on the NCI-60 and Bursi datasets. The predictive power of the other kernels (PDK and WDK) is also much improved when working with the augmented graphs. The performance gap with respect to more complex and expensive kernels is significantly reduced.

2.6 Conclusions

The major contribution of this chapter is the introduction of a simple but effective way to incorporate background knowledge in graph-based representations of molecular data. To demonstrate the effectiveness of the proposed approach, we tested several graph kernel models on the augmented representations. Moreover, the proposed technique has been tested on a wide range of different types of chemoinformatics classification tasks. In all cases we observe a consistent improvement of the predictive performance. Finally, when providing the background knowledge to the PMCSK, we found that it significantly outperforms the current state-of-the-art algorithm on the NCI-60 dataset in 28 of the 60 cell lines, with the other 32 cell lines showing no significant difference in accuracy, and it obtains near-optimal results on the Bursi mutagenicity task.

The software for augmentation, including DMax Chemistry Assistant, is available at no cost at `http://dtai.cs.kuleuven.be/dmax/` .

Chapter 3

Neighborhood Subgraph Pairwise Distance Kernel

In this chapter, we introduce a novel graph kernel called the Neighborhood Subgraph Pairwise Distance Kernel. The kernel decomposes a graph into all pairs of neighborhood subgraphs of small radius at increasing distances. We show that using a fast graph invariant we obtain significant speed-ups in the Gram matrix computation. Finally, we test the novel kernel on a wide range of chemoinformatics tasks, from antiviral to anticarcinogenic to toxicological activity prediction, and observe competitive performance when compared against several recent graph kernel methods.

This chapter is based on (Costa and De Grave 2010) and (De Grave and Costa 2010a).

3.1 Introduction

Since the introduction of convolution kernels in (Haussler 1999), the decomposition approach has been the the guiding principle in kernel design for structured objects. Recall from Section 2.3.2 that, according to this approach, a similarity function between discrete data structures can be obtained by decomposing each object into parts and by devising a valid local kernel between the subparts. For over ten years, machine learning researchers have exploited the remarkable property that it is possible to efficiently compute this type of kernels even when objects admit an exponential number of decompositions. This is possible if an efficient method to enumerate the parts can be produced and if the sum over a potentially exponential number of local kernel evaluations can be performed in polynomial time (e.g. through dynamic programming). However, as the dimension of the feature space associated with the kernel becomes exponentially larger, there is an increasing probability that a significant fraction of the feature space dimensions will be poorly correlated with the target function. As a consequence, even when using large margin classifiers, one can fail to obtain models with good generalization performance (Ben-David et al. 2002). Possible remedies include down-weighting the contribution of larger fragments and/or bounding a priori their size. Alternatively one can try to identify a strong bias, relevant to the task at hand, and consider only a selected subset of structures to limit the dimension of the feature space without degrading the prediction performance. Here we limit the extracted substructures by design, following the physico-chemical intuition that the full molecule's behavior is contained in its electron density field. As we saw in the previous chapter, the continous field can be discretized using the notion of *functional group*, empirically discovered by chemists since a long time. In essence, a functional group is a specific molecular subgraph which can be viewed as characteristic local electron density distribution that remains fairly constant, independent of the environment. In the same spirit, in this chapter we employ pairs of neighborhood subgraphs of increasing sizes (i.e. subgraphs induced by all "nearby" vertices, see Section 1.2). Since each vertex in a molecular graph gives rise to a constant, small number of such subgraphs, the neighborhood graphs can be efficiently enumerated in linear time. In Section 3.3 we show how to perform quick equality matches between large neighborhood subgraphs, which allows us to obtain very fast Gram matrix computation runtimes. We empirically verify in Section 3.7 that the proposed approach yields predictive models with competitive performance on a wide range of bio- and chemoinformatics tasks.

3.2 Neighborhood Subgraph Pairwise Distance Kernel

In the following we define the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) as an instance of a decomposition kernel (see Section 2.3.2). We will again use the notation introduced in Chapter 1. In particular, recall that a neighborhood subgraph Ω_r^v is the subgraph induced by the neighborhood of radius r of v and rooted in v (Definition 1.2.13).

We define the relation $R_{r,d}(A_v, B_u, G)$ between two rooted graphs A_v, B_u and a graph G to select all pairs of neighborhood subgraphs of radius r whose roots are at distance d in a given graph G:

$$R_{r,d}(A_v, B_u, G) \Leftrightarrow (G \in \mathcal{G} \wedge A_v, B_u \in \{\Omega_r^z : z \in V_G\} \wedge d(u,v) = d) \qquad (3.2.1)$$

We define $\kappa_{r,d}$ over $\mathcal{G} \times \mathcal{G}$ as the decomposition kernel on the relation $R_{r,d}$, that is:

$$\kappa_{r,d}(G, G') = \sum_{\substack{\{A_v, B_u | R_{r,d}(A_v, B_u, G)\} \\ \{A'_{v'}, B'_{u'} | R_{r,d}(A'_{v'}, B'_{u'}, G')\}}} \delta(A_v, A'_{v'})\delta(B_u, B'_{u'}) \qquad (3.2.2)$$

where the *exact matching kernel* $\delta(x, y)$ is 1 if $x \simeq y$ (i.e. if the labeled graph x is isomorphic to y) and 0 otherwise. In words: $\kappa_{r,d}$ counts the number of identical pairs of neighborhood graphs of radius r at distance d between two graphs (see Figure 3.1).

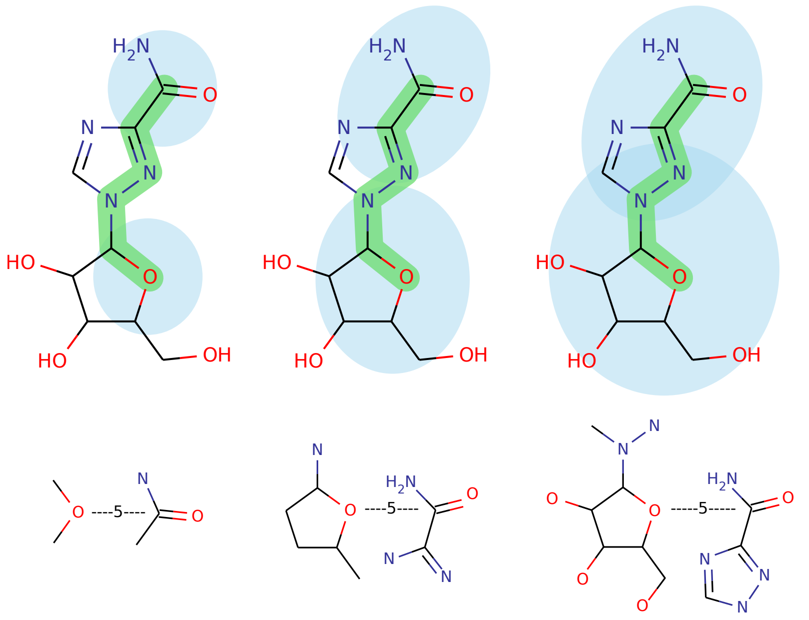


Figure 3.1: Illustration of pairs of neighborhood graphs for radius $r = 1, 2, 3$ and distance $d = 5$. Note that neighborhood graphs can overlap.

The Neighborhood Subgraph Pairwise Distance Kernel is finally defined as:

$$K(G, G') = \sum_r \sum_d \kappa_{r,d}(G, G'). \qquad (3.2.3)$$

For efficiency reasons however, in this work we consider the feature zero-extension of K obtained by imposing an upper bound on the radius and the distance parameter:

$$K_{r^*,d^*}(G, G') = \sum_{r=0}^{r^*} \sum_{d=0}^{d^*} \kappa_{r,d}(G, G') \tag{3.2.4}$$

That is, we are limiting NSPDK to the sum of the $\kappa_{r,d}$ kernels for all increasing values of the radius (distance) parameter up to a maximum given value r^* (d^*). Furthermore we consider a normalized version of $\kappa_{r,d}$, that is:

$$\hat{\kappa}_{r,d}(G, G') = \frac{\kappa_{r,d}(G, G')}{\sqrt{\kappa_{r,d}(G, G)\kappa_{r,d}(G', G')}} \tag{3.2.5}$$

to ensure that relations of all orders are equally weighted regardless of the size of the induced part sets[1].

Finally, it is easy to show that the Neighborhood Subgraph Pairwise Distance Kernel is a valid kernel as: 1) it is built as a decomposition kernel over the countable space of all pairs of neighborhood subgraphs of graphs of finite size; 2) the kernel over parts (the exact matching kernel) is a valid kernel; 3) the zero-extension to bounded values for the radius and distance parameters preserves the kernel property; and 4) so does the normalization step.

3.3 Graph invariant

The NSPDK includes an exact matching kernel over two graphs which is equivalent to solving the graph isomorphism problem (ISO). Since the existence of (deterministic) polynomial algorithms for ISO is still an open problem, we have to resort to either of two strategies:

- limit the class of graphs under consideration and solve ISO exactly; or
- give an approximate (fast) solution of ISO on general graphs.

The former option is viable for molecules because they have bounded degree, in which case a polynomial algorithm exists (Luks 1982). Here we opt for the latter solution since we are mainly concerned with application domains

[1]As the number of neighborhood graphs grows exponentially with the radius, large (infrequent) subgraphs tend to dominate the kernel value with negative effects on the generalization performance of predictive systems.

where the number of graphs to be processed is very large[2] and application specific pre-processing might alter the class of the input graphs. For example, molecular graph augmentation as discussed in Chapter 2 makes the graphs non-outerplanar, which is why we had to present separate atom-bond and moiety graphs to the PMCSK.

We implement an approximation of the exact matching kernel $\delta(G_h, G'_{h'})$ in two steps:

1. We compute a fast graph invariant encoding for G_h and $G'_{h'}$ via a label function $\lambda^g : \mathcal{G}_h \rightarrow \Sigma^*$, where $\mathcal{G}_h$ is the set of rooted graphs and Σ^* is the set of strings over a finite alphabet Σ.
2. We make use of a hash function $H : \Sigma^* \rightarrow \mathbb{N}$ to confront $H(\lambda^g(G_h))$ and $H(\lambda^g(G'_{h'}))$.

In other words, we produce an efficient string encoding of graphs from which we obtain a unique identifier via a hashing function from strings to natural numbers. In this way, the isomorphism test between two graphs is reduced to a fast numerical identity test. Note that we cannot hope to exhibit an efficient certificate for isomorphism in this way, but only an efficient graph invariant at most, i.e. there will be cases where two non-isomorphic graphs are assigned the same identifier.

The graph encoding $\lambda^g(G_h)$ that we propose is best described by introducing new label functions for vertices and edges, denoted λ^n and λ^e respectively. $\lambda^n(v)$ assigns to vertex v the concatenation of the lexicographically sorted listed of distance-label pairs $(d(v,u), \lambda(u))$ for all $u \in G_h$. Since G_h is a rooted graph we can exploit the knowledge about the identity of the root vertex h and include, for each vertex v, the additional information of the distance from the root node $d(v,h)$. $\lambda^e(uv)$ assigns to edge uv the label $(\lambda^n(u), \lambda^n(v), \lambda(uv))$. $\lambda^g(G_h)$ assigns to the rooted graph G_h the concatenation of the lexicographically sorted list of $\lambda^e(uv)$ for all $uv \in E_{G_h}$. In other words, we relabel each vertex with a string that encodes the vertex distance from all other labeled vertices (plus the distance from the root vertex); the graph encoding is obtained as the sorted edge list, where each edge is annotated with the endpoints' new labels. Algorithm 1 shows pseudocode for computing the invariant and Figure 3.2 illustrates the computation for a concrete graph.

The label function $\lambda^g(G_h)$ only needs to operate on connected graphs, for a neighborhood subgraph is always connected. We will now sketch why $\lambda^g(G_h)$ is a graph invariant.

[2]The ZINC database lists the structure of over 18 million chemical compounds that are commercially available. Over 13 million of these are drug-like.

Algorithm 1 Rooted graph invariant $\lambda^g(G_h)$

```
1: Given: graph $G = (V, E)$, root $h \in G$, vertex and edge label function $\lambda$
2: Return: rooted-graph invariant string $\lambda^g(G_h)$
3:
4: Compute all distances $d(u, v)$
5: for all $u \in V$ do
6:     $\mathbb{D} \leftarrow multiset\{(d(u, v), \lambda(v))|v \in V\}$
7:     $\lambda^n(u) \leftarrow concat(d(u, h), sort(\mathbb{D}))$
8: end for
9: for all $uv \in E$ do
10:    $\lambda^e(uv) \leftarrow concat(\lambda^n(u), \lambda^n(v), \lambda(uv))$
11: end for
12: $\mathbb{E} \leftarrow multiset\{\lambda^e(uv)|uv \in E\}$
13: $\lambda^g(G_h) \leftarrow concat(sort(\mathbb{E}))$
14: return $\lambda^g(G_h)$
```

Lemma 3.3.1. *Isomorphism preserves distances. Given an isomorphism ϕ and two isomorphic simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$:*

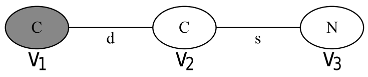
$$\forall u, v \in V : d(u, v) = d(\phi(u), \phi(v))$$

Proof. (Sorlin and Solnon 2004, page 292) If ϕ is an isomorphism, then $uv \in E_1 \Leftrightarrow \phi(u)\phi(v) \in E_2$. Therefore, $v_0, v_1, \ldots, v_n$ is a path in G_1 iff $\phi(v_0), \phi(v_1), \ldots, \phi(v_n)$ is a path in G_2. Thus, $v_0, v_1, \ldots, v_n$ is a shortest path in G_1 iff $\phi(v_0), \phi(v_1), \ldots, \phi(v_n)$ is a shortest path in G_2. Hence, the length of the shortest path(s) between v_1 and v_n equals the length of the shortest path(s) between $\phi(v_1)$ and $\phi(v_n)$. $\square$

Since the distances are preserved (Lemma 3.3.1) and also the labels (Definition 1.2.15), we find that the multisets[3] $\mathbb{D}_1$ and $\mathbb{D}_2$ (see Algorithm 1 line 6) are equal for isomorphic graphs G_1 and G_2. The isomorphism will then also preserve $\lambda^n(v)$. Note that the lexicographic sorting is essential for the graph invariance: we cannot concatenate the multiset in an arbitrary order and expect the strings to be equal. By Definition 1.2.15, also $\lambda(uv)$ will be preserved, hence also $\lambda^e(uv)$. The multisets $\mathbb{E}_1$ and $\mathbb{E}_2$ are thus equal for isomorphic graphs, and so are their sorted concatenations.

Finally, we resort to a Merkle-Damgård construction based hashing function for variable-length data to map the graph encoding string to a fixed-length integer. For additional performance and to reduce memory use, the intermediate $\lambda^n(u)$

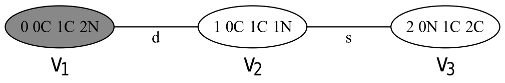
[3]A multiset or bag can contain multiple instances of the same element.



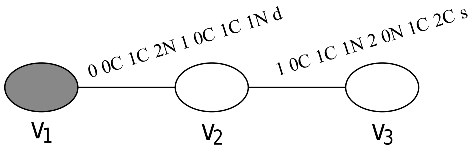
(a) Line 1: Labeled input graph

	v_1	v_2	v_3
v_1	0	1	2
v_2	1	0	1
v_3	2	1	0

(b) Line 4: Compute all distances



(c) Line 7: Relabel vertices



(d) Line 10: Relabel edges

$$\lambda^g(G_h) = \text{0 0C 1C 2N 1 0C 1C 1N d . 1 0C 1C 1N 2 0N 1C 2C s}$$

(e) Line 13: Output sorted edge labels

Figure 3.2: Example computation of the rooted graph invariant $\lambda^g(G_{v_1})$. Line numbers refer to algorithm 1.

values are also hashed. It is trivial to control the size of the feature space by choosing the hash codomain size (or alternatively the size in bits of the returned hashed values)[4]. Reduced-size feature spaces may be useful for very-large-scale applications where memory and time is of great concern, but for the moderately sized data sets in this chapter we used 32-bit encoding of neighborhood pairs with satisfactory performance (Table 3.2).

[4]Naturally there is a tradeoff between the size of the feature space and the number of hash collisions.

3.4 Algorithmic complexity

The time complexity of the NSPDK depends on two key procedures:

1. the extraction of all pairs of neighborhood graphs Ω_r^v at distance $d = 0, \ldots, d^*$, and
2. the computation of the graph invariant for those subgraphs.

The first procedure can be efficiently implemented by factoring it into a) the extraction of Ω_r^v for all $v \in V_G$ and b) the computation of distances between pairs of vertices whose pairwise distance is less than d^*. For this latter step we can repeat a breadth-first (BF) visit up to distance d^* for each vertex in $O(|V_G||E_G|)$. Note that, on graphs with bounded (low) degree, the complexity is more realistically modeled as a linear function of $|V_G|$ since a small d^* implies, in practice, that each bounded BF visit can be performed in constant time. The complexity of point a) is linear in the number of edges in the neighborhood (constant in practice for small r).

The complexity of point 2 (the computation of the graph invariant for neighborhood graphs) can be analyzed in terms of i) the computation of the string encoding $\lambda^g(G_h)$ and ii) the computation of the hash function $H(\lambda^g(G_h))$. Part i) is dominated by the computation of all pairwise distances in $O(|V_{G_h}||E_{G_h}|)$ and the sorting of the relabeled edges, which has complexity $O(|V_{G_h}||E_{G_h}| \log |E_{G_h}|)$ since edges are relabeled with strings containing the distance information of the endpoints from all other vertices. The hash function complexity (part ii)) is linear in the size of the string.

We conclude that the overall complexity

$$O(|V_G||V_{G_h}||E_{G_h}| \log |E_{G_h}|) \tag{3.4.1}$$

is dominated by the repeated computation of the graph invariant for each vertex of the graph. Since this is a constant time procedure for small values of d^* and r^*, we conclude that the complexity of NSPDK is in practice linear in the size of the graph.

Note finally that, to reduce space complexity, we do not manage the hash collisions, as this would force the algorithm to keep in memory all the encoding key - hashed value pairs.

3.5 Related work

The NSPDK combines in a kernel fashion ideas present in two popular chemoinformatics fingerprint methods: the *circular substructure* and the *atom pair representation*.

A circular substructure representation encodes the immediate neighborhood of an atom. It does so by assigning an initial code to an atom based on the atom type and other information such as the number of bonds, the electric charge, donor/acceptor tendency, etc. The code for an atom and all its neighborhood is then hashed to produce second order encodings. The process is then iterated a given number of times k. The order k corresponds to the radius in bonds up to which features are generated and typically $k = 1$ or 2. Popular descriptors of this type are the Extended Connectivity Fingerprints (ECFP) and the Functional Connectivity Fingerprints developed at SciTegic (now Accelrys) which have been shown to be effective in similarity search operations (Hert et al. 2004).

The atom pair representation is an adaptation of the pharmacophore points technique to the 2D rather than 3D structural representation. Here all pairs of atoms are encoded together with the length of the shortest path between them. Each atom is typically described by its type and the number of non-hydrogen atoms to which it is bonded or in terms of its binding properties such as being a cation, anion, neutral, hydrogen bond donor/acceptor, hydrophobic, etc . Popular descriptors of this type are the CATS (Chemically Advanced Template Search) (Schneider et al. 1999) and the Similog keys (Schuffenhauer et al. 2003) where the number of occurrences of a particular pair (rather than its presence or absence as it is more usual in conventional fingerprint representations) is used. Typically only pairs for distances up to 10 bonds are used (Hert et al. 2004).

3.6 Empirical evaluation

We primarily want to answer two questions about the proposed kernel:

Q1 How does the generalization performance of NSPDK compare to other recent graph kernels?

Q2 How does the experimental runtime of NSPDK compare to other fast graph kernels?

3.6.1 Datasets

We will reuse all four datasets introduced in Section 2.4.1: NCI-60, HIV, PTC, and Bursi. To supplement these small molecule datasets, we add one dataset that contains of a different type of graph.

D&D

The D&D dataset of 1178 protein structures was constructed by (Dobson and Doig 2003) and transformed by (Shervashidze and Borgwardt 2009) into a problem of binary classification of graphs, where the task is to distinguish enzymes from non-enzymes. Each protein has been converted into a graph, considering the amino acids as nodes[5] and considering two nodes linked if their 3D distance in the folded protein is less than 6 ångströms. Note that, while small molecules induce graphs with $\approx$ 30 nodes, protein graphs result in much larger graphs ($\approx$ 300 nodes), with some instances exhibiting several thousands of vertices.

3.6.2 Benchmarking graph kernels

In the following we briefly describe several other graph kernels that will be used for benchmarking the proposed approach. In particular we restrict our attention to kernels that do not decompose the graph in walks or paths as it has been shown in (Menchetti et al. 2005) and (Shervashidze and Borgwardt 2009) that they tend to exhibit lower accuracy and have higher runtimes. Here we consider the Graph Fragment Kernel (GFK) introduced in (Wale et al. 2008), the Weighted Decomposition Kernel (WDK) by (Menchetti et al. 2005), the Pairwise Maximum Common Subgraphs Kernel (PMCSK) introduced in (Schietgat et al. 2009), the Neighborhood Subgraph Kernel (NSK) similar in spirit to the fast kernel presented in (Shervashidze and Borgwardt 2009), and the Pairwise Distance Kernel (PDK), similar to the Equal Length Shortest-Path Kernel by (Borgwardt and Kriegel 2005).

GFK The GFK feature space is obtained considering all connected subgraphs up to a given maximum number of edges. GFK differs from NSPDK as it considers an unbiased (i.e. all possible) type of subgraph rather than neighborhood subgraphs. Note that GFK induces larger explicit representations

[5]The node label alphabet has size $\approx$ 90 rather than 20 as the various types of ambiguities are explicitly encoded as additional labels.

of molecular graphs even when limiting the subgraph size to relatively small values: on average more than 516 $\pm$ 381 features per molecule are generated when allowing subgraphs with less than 8 edges on the NCI-60 dataset. NSPDK, in a comparable setting, generates 28 $\pm$ 9 different neighborhood subgraphs per molecule, yielding 251 $\pm$ 143 features when considering subgraph pairs up to maximum distance 5.

WDK In the WDK (Section 2.3.3) the neighborhood of a given radius is first associated to each vertex in a graph. The WDK is then computed as the product of an exact matching kernel over the vertex label with a kernel over the neighborhood edge multiset. Here the edge label information is augmented with the endpoints labels. Among the differences between WDK and NSPDK there are: the single vs. pairwise subgraph approach, and the "soft" similarity match vs. the "hard" isomorphism match of neighborhood subgraphs.

PMCSK The PMCSK (Section 2.3.4) feature space is obtained considering the maximum common subgraph (MCS) between all pairs of instances in the training set. The authors show that, although the computation of the maximum common subgraph in the general case is an NP-hard problem, one can employ a polynomial-time algorithm if only outerplanar graphs are considered in combination with a special case of subgraph isomorphism called block-and-bridge-preserving (BBP) subgraph isomorphism (Schietgat et al. 2008). In addition to the pairwise vs. single subgraph approach, PMCSK differs from NSPDK in the specific type of subgraphs considered (MCSs vs. neighborhood graphs).

NSK In the NSK the feature space is obtained considering the neighborhood subgraphs of increasing radii up to a maximum radius r^*. The NSK features are similar in spirit to those obtained by the circular substructure approach (see Section 3.5).

PDK The PDK (Section 2.3.1) computes the similarity between two graphs by comparing all pairs of vertices annotated with their pairwise distance. We consider the zero-extension of PDK up to a maximum distance d^*. The PDK features are similar in spirit to those obtained by the atom pair representation approach (see Section 3.5).

The NSK and the PDK are special cases of NSPDK: NSK is obtained considering the NSPDK with a maximum distance $d^* = 0$ while PDK is obtained considering the NSPDK with a maximum radius $r^* = 0$. The optimal value for the remaining free parameter (r^* for NSK and d^* for PDK) is experimentally determined via cross-validation.

3.6.3 Empirical properties of the NSPDK

We measured the size of the neighborhood graphs used in the computation of NSPDK with radius ranging between 1 and 4 for the NCI-60 dataset, obtaining 3, 6, 10 and 13 vertices respectively (and approximately the same values for the edge count). We observe that NSPDK can consider significantly larger subgraphs if compared to the GFK (7 edges) with comparable runtimes (see Section 3.7).

We tested how well the graph invariant proposed in Section 3.3 approximates an isomorphism certificate on chemical graphs. On subgraphs extracted from the NCI-60 dataset the approximation is perfect: there are no non-isomorphic subgraphs in the set that received the same identifier. The exact isomorphism test has been computed via the VFLib Graph Matching Library[6].

We computed the number of hash collisions when encoding pairs of neighborhood subgraphs at distances ranging from 0 to 10, on the NCI-60 dataset: for neighborhood subgraphs of radius 2 we do not have collisions, for radius 3 we have 2 collisions out of 551198 unique pairs and for radius 4 we have 15 collisions out of 667505 unique pairs. We conclude that the error introduced by hashing collisions is in practice negligible.

3.6.4 Experimental setup

We tested the predictive performance of SVM-Light (Joachims 1999) by stratified 10-fold cross-validation, keeping folds identical between kernels.

We evaluated the generalization performance of the kernels and the augmentation method by the area under the receiver operating characteristic (AUROC) (the plot of the fraction of true positives versus the fraction of false positives).

A number of parameters were optimized by internal cross-validation on the Bursi data. We allowed each kernel to be optionally composed with a polynomial kernel of degree 3, 5, or 7, or with an RBF kernel with gamma equal to 0.1. The trade-off between training error and margin (c in Equation 1.4.7) was selected from $\{1,10,100\}$. The maximum radius r^* was selected from $\{0,1,2,3,4,5\}$, and maximum distance d^* from $\{3,4,5,6,8,10,12,14,20\}$. Augmented

[6]http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html

and unaugmented graphs were optimized separately. The most frequently selected parameter values for each kernel as found in Bursi were then used for the other datasets, except for D&D where they were optimized separately due to the different nature of the data. For D&D, only radii up to three were considered.

All kernels were normalized before composition, using Equation 1.3.7. The cost factor (-j) was set to the prevalence ratio of negative to positive examples in the training set. All other SVM-Light parameters were left at their default value.

We used a radius of 4 for the WDK, motivated by the results on the HIV dataset by (Menchetti et al. 2005).

For GFK we used the AFGen program of (Wale et al. 2008) to obtain the feature vector. The maximum subgraph size was set to the default value of 7 edges. We did not implement the length-differentiated min-max kernel but rather used the same extensive kernel parameter optimization as for all other kernels. We could not run AFGen on augmented graphs because these exceed its hard-coded maximum node degree.

The same code, with the appropriate parameter settings, has been used for the NSPDK, NSK and PDK.

PMCSK was not optimized as described above, but used a Tanimoto kernel, no cost factor, and internally cross-validated values for c as in (Schietgat et al. 2009).

Composition with a third degree polynomial was chosen for all kernels except for unaugmented PDK and WDK which used fifth degree, and augmented NSPDK which used a linear kernel. The parameter c was always 1 except for augmented NSPDK where it was 10. The optimal choice for r^* was 3, except for augmented NSK where 2 was sufficient. There was some variation in the choice of d^*: 12 for unaugmented PDK, 3 for augmented PDK, 5 for unaugmented NSPDK and 4 for augmented NSPDK.

3.7 Results and Discussion

In Table 3.1, we present an overview of the AUROC performance for an SVM model trained with different graph kernels over unaugmented and augmented molecular graphs (denoted G and G_{aug} respectively). In the same table we indicate in boldface the methods with highest *accuracy*, or not significantly worse according to the binomial sign test at $p < 0.05$. We observe that the proposed NSPDK is never significantly worse than the most accurate method. We note moreover that NSPDK performance compares favorably with the best results reported in (Wale et al. 2008) (the state-of-the-art on HIV and NCI-60 to the authors' knowledge). As a side note, we report that the relative error reduction, when using the augmented molecular graphs vs. the unaugmented ones, varies from 5% for PDK, to 3% for WDK, to 1% for PMCSK, NSK and NSPDK; this result is in agreement with the intuition that graph methods sensible to larger fragments automatically capture most of the functional group related information. The error rates of augmented NSK and NSPDK on Bursi are about 14.5%. Unaugmented NSPDK and augmented PMCSK achieve 15%. Since (Kazius et al. 2005) mention that the average interlaboratory reproducibility error of Ames tests is 15%, one cannot hope, on this dataset, to do much better.

Results for the D&D dataset have been computed only for NSK and NSPDK due to infeasible runtimes for WDK and PMCSK, and unmet hard-wired constraints on the vertex labels and degree for GFK. NSK achieved AUROC 81.4% with radius 0 (and around 80% with higher radii), while NSPDK achieved 85.9% with radius 1 and maximum distance 3 (85.5% with higher radii). The runtimes for the D&D Gram matrix computation were $7.9 \cdot 10^2$ and $8.2 \cdot 10^2$ seconds respectively. We observe that NSPDK achieves a 30.2% error reduction over the best results reported by (Shervashidze and Borgwardt 2009) with comparable runtimes.

In Table 3.2 we report the runtime required for the Gram matrix computation for the different kernels, normalized but not composed. Augmented molecular graphs have a significantly larger number of vertices and edges, hence are slower to process. The time required for augmentation is shown separately. Obviously, the augmentation step is of linear time complexity in the number of molecules, while the Gram matrix computation is quadratic. The programs were executed on a single core of an Intel Core2 Quad Q9550 CPU (2.83GHz), except for the HIV dataset which was run on an Intel Xeon E5420 CPU (2.5GHz) due to 64-bit support of the operating system.

We observe that:

- The runtime for the WDK neighborhood soft matching is one order of magnitude higher than the graph invariant identity test for NSPDK.
- The runtime for extracting and matching maximum common subgraphs for all pairs of molecules in PMCSK is three orders of magnitude higher than the graph invariant extraction and identity test for NSPDK.
- Runtimes for random walk kernels and tree kernels are, as reported in (Shervashidze and Borgwardt 2009), five to six orders of magnitude higher (estimated on two NCI datasets) than for NSPDK.

Table 3.1: Generalization performance of kernels on unaugmented and augmented molecular graphs

	NCI-60 (avg.)	HIV CA vs. CM	HIV CACM vs. CI	HIV CA vs. CI	PTC (avg.)	Bursi
AUROC (%)						
GFK(G)	77.8 ± 2.3	82.0 ± 4.7	82.8 ± 1.9	93.9 ± 2.6	62.6 ± 10	89.6 ± 0.3
WDK(G)	71.1 ± 2.4	83.1 ± 4.3	82.9 ± 1.8	94.0 ± 3.4	62.1 ± 7.7	88.0 ± 0.4
WDK(G_{aug})	**80.0 ± 2.3**	84.2 ± 4.3	83.9 ± 1.7	95.0 ± 2.7	65.1 ± 8.7	90.8 ± 0.2
PMCSK(G)	79.6 ± 2.2	**82.6 ± 6.2**	81.8 ± 2.2	93.0 ± 3.7	64.5 ± 8.8	90.5 ± 1.3
PMCSK(G'_{aug})	**80.3 ± 2.2**	**82.8 ± 6.2**	**83.2 ± 2.1**	93.4 ± 3.4	65.6 ± 8.8	**91.5 ± 1.1**
PDK(G)	73.4 ± 2.6	81.6 ± 4.6	77.7 ± 1.9	92.6 ± 3.2	61.2 ± 9.7	82.7 ± 0.3
PDK(G_{aug})	77.8 ± 2.4	82.1 ± 4.2	83.4 ± 2.1	94.5 ± 2.4	64.6 ± 9.9	89.3 ± 0.3
NSK(G)	79.1 ± 2.2	**84.2 ± 4.9**	84.3 ± 2.0	95.3 ± 1.5	67.4 ± 9.4	91.6 ± 0.2
NSK(G_{aug})	79.4 ± 2.2	**84.4 ± 4.5**	84.1 ± 2.2	94.9 ± 2.1	67.1 ± 9.3	**91.8 ± 0.2**
NSPDK(G)	79.5 ± 2.2	**83.9 ± 5.6**	83.8 ± 2.1	**95.6 ± 1.3**	**69.3 ± 9.5**	**91.7 ± 0.3**
NSPDK(G_{aug})	**80.1 ± 2.2**	**84.1 ± 4.8**	**84.9 ± 2.1**	**95.1 ± 2.0**	**68.9 ± 9.8**	**92.0 ± 0.2**

Table 3.2: Net CPU time of graph kernels in seconds

	NCI-60	HIV	PTC	Bursi
Number of mol.	3910	42687	417	4337
Aug. time	$3.5 \cdot 10^2$	$3.4 \cdot 10^3$	$3.4 \cdot 10^1$	$1.2 \cdot 10^2$
GFK(G)	$3.5 \cdot 10^1$	$1.4 \cdot 10^4$	$3.1 \cdot 10^0$	$7.3 \cdot 10^1$
WDK(G)	$1.8 \cdot 10^3$	$1.6 \cdot 10^5$	$8.0 \cdot 10^0$	$1.1 \cdot 10^3$
WDK(G_{aug})	$2.3 \cdot 10^3$	$2.3 \cdot 10^5$	$1.4 \cdot 10^1$	$1.5 \cdot 10^3$
PMCSK(G)	$2.8 \cdot 10^5$	$3.3 \cdot 10^{4*}$	$6.2 \cdot 10^2$	$3.5 \cdot 10^5$
PMCSK(G'_{aug})	$2.8 \cdot 10^5$	$3.3 \cdot 10^{4*}$	$6.3 \cdot 10^2$	$3.5 \cdot 10^5$
PDK(G)	$4.2 \cdot 10^1$	$3.9 \cdot 10^3$	$1.0 \cdot 10^0$	$3.6 \cdot 10^1$
PDK(G_{aug})	$7.7 \cdot 10^1$	$4.2 \cdot 10^3$	$2.0 \cdot 10^0$	$5.7 \cdot 10^1$
NSK(G)	$6.2 \cdot 10^1$	$3.1 \cdot 10^3$	$2.8 \cdot 10^0$	$5.1 \cdot 10^1$
NSK(G_{aug})	$3.5 \cdot 10^2$	$6.0 \cdot 10^3$	$1.4 \cdot 10^1$	$2.0 \cdot 10^2$
NSPDK(G)	$1.2 \cdot 10^2$	$1.0 \cdot 10^4$	$3.4 \cdot 10^0$	$1.1 \cdot 10^2$
NSPDK(G_{aug})	$4.6 \cdot 10^2$	$1.9 \cdot 10^4$	$1.6 \cdot 10^1$	$2.9 \cdot 10^2$

* MCSs derived only from the 1503 CA-CM molecules.

3.8 Conclusions

In this chapter, we presented a novel and fast graph kernel based on exact matching between pairs of small subgraphs. Empirical results confirm the intuition that using relatively large fragments in a pairwise fashion improves generalization performance on a wide range of bio- and chemoinformatics tasks. Moreover, the use of fast graph invariant procedures allows a speed-up of several orders of magnitude for Gram matrix computations when compared with kernels based on soft matching or more complex subgraph definition.

The source code of the kernel can be obtained from `http://dtai.cs.kuleuven.be/ml/systems`.

Chapter 4

kLog: a Language for Logical and Relational Learning with Kernels

Having observed both excellent generalization capabilities and fast kernel evaluation using NSPDK for prediction tasks for molecular graphs (Chapter 3), one may wonder whether the technique can also be applied to machine learning problems outside the domain of molecular graphs.

This question has inspired a system, called *kLog*, that permits to apply the principles and infrastructure of graph kernel based learning to more general data: relational databases. This is achieved through a process called *graphicalization*: the transformation of relational representations into graphs. In the current chapter, we will briefly describe the main concepts and the high-level design of the kLog system. The chapter is based on a homonymous draft paper by Paolo Frasconi, Fabrizio Costa, Luc De Raedt, and the author of this thesis (Frasconi et al. 2011).

The NSPDK itself is included as one of the default kernels in kLog. After all, NSPDK is a generic graph kernel and thus can operate on arbitrary graphs. Whether its feature set of neighborhood subgraph pairs forms a suitable learning bias, depends on the property to be learned and on the input graphs. One can expect better results when the input graphs are molecule-like. In particular, molecular graphs have a bounded degree, and the number of vertices one can reach with increasing distance from any starting node, rises only gradually with the distance. In Section 4.3.2, we will propose modifications that make the kernel suitable for a broad class of graphs.

4.1 Relational learning with kernels

The field of statistical relational learning (SRL) is populated with a fairly high number of models and alternative representational approaches, a state-of-affairs often referred to as the "SRL alphabet soup" (Dietterich et al. 2008, De Raedt et al. 2008). Even though there are many differences between these approaches, they typically define a model as a probability distribution over possible worlds or interpretations (explained in Section 4.2). In the machine learning literature (De Raedt 2008b), interpretations are often used to model relational learning problems because they naturally support the modeling of entities as well as the relationships amongst them. It is also this representation that is adopted in kLog.

However, unlike typical statistical relational learning frameworks, kLog does not employ a probabilistic framework but is rather based on linear modeling. We assume that interpretations are sampled identically and independently from a fixed distribution. They are represented as input-output pairs $z = (x, y)$ where x and y are sets of facts. The task is to a learn a function that will map the inputs to the output.

To construct a model, a feature vector $\phi(x, y)$ is first associated with each interpretation. A linear model $F(x, y) = w^\top \phi(x, y)$ is then used to score the interpretation. Prediction is the process of maximizing F with respect to y. Learning is the process of fitting w to the available data, typically using some statistically motivated loss function that measures the discrepancy between the prediction $f(x_i) = \arg\max_y F(x_i, y)$ and the observed output y_i on the i-th training instance.

Linear modeling on features covers a number of commonly used algorithms. Support vector machines (SVM, see Section 1.4) optimize for small $\|w\|_2$ plus the hinge loss function. That is, examples on the correct side of the margin do not cause any loss, and the remaining examples cause a loss linear to their distance to the correct margin. Logistic regression maximizes the conditional likelihood of outputs given inputs, which can be seen as minimizing a smoothed version of the SVM hinge loss. Naive Bayes maximizes the joint likelihood of inputs and outputs.

4.2 Data modeling

kLog builds upon logical and relational data representation, which is closely related to the classic entity-relationship (E/R) data model, a commonly used design tool in database development, see e.g. (Garcia-Molina et al. 2009). The main ontological assumption is that the domain can be described by objects and relations. Relations are also called predicates. Being based on clausal logic, kLog employs the notation and the semantics underlying the programming language Prolog and the database formalism Datalog.

kLog learns from *interpretations*. An interpretation is essentially a set of ground atoms. A ground atom $r(c_1, ...c_n)$ is a relation symbol r of arity n followed by an n-tuple of constants c_i. The relation symbol r is also known as the predicte name. We denote by $\mathcal{C}$ the set of constants (objects) in the domain and by $\mathcal{R}$ the set of relations.

An interpretation contains all the atoms that are *true* and all atoms not in the interpretation are assumed to be *false*. It can therefore also be regarded as a set of tuples in a relational database. So, in database terminology, interpretations correspond to instances of a relational database. In logic programming terminology, we are using so-called Herbrand interpretations.

Because of the close similarity between the semantics of kLog and databases, we will borrow a few terms from database theory. For example, we will use the term *column* to refer to the set of constants that, in a given intepretation, appear as the i-th argument in true ground atoms of a certain relation.

kLog makes some assumptions on valid interpretations. In particular:

A.1 (*Object types*). The set of constants $\mathcal{C}$ is partitioned into a set of *entity identifiers* $\mathcal{E}$ (or identifiers for short) and a set of *property values* $\mathcal{V}$). Identifiers are themselves partitioned into k subsets $\mathcal{E}_1, \ldots, \mathcal{E}_k$ called *entity-sets*.

A.2 (*Finiteness*). The interpretations are *finite* sets of atoms.

A.3 (*Typed relations*). For every relation, every column consists of either property values or identifiers from one particular entity-set. The type signature for a relation $r/m \in \mathcal{R}$ is an expression of the form

$$r(\mathsf{role}_1 :: \mathsf{type}_1, \ldots, \mathsf{role}_m :: \mathsf{type}_m)$$

where, for all $j = 1, \ldots, m$, $\mathsf{type}_j \in \{\mathcal{E}_1, \ldots, \mathcal{E}_k, \mathcal{V}\}$ and role_j is the role of the j-th column of r. The type signature is closely related to the type declarations that are typical in logical and relational learning.

A.4 (*Keys*). The primary key of every relation consists of the columns whose type belongs to $\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$. The *relational arity* of a relation is the length of its primary key. As a special case, relations of zero relational arity are admitted and they must consist of at most a single tuple in any interpretation[1].

A.5 (*E-relations and R-relations*). For every entity-set $\mathcal{E}_i$ there is a distinguished relation $r/m \in \mathcal{R}$ such that: r/m has relational arity 1 and has a key of type $\mathcal{E}_i$. These distinguished relations are called *E-relations* and describe entity-sets, possibly with $(m-1)$ attached properties as satellite data. The remaining $|\mathcal{R}| - k$ relations are called *R-relations* or *relationships*[2]. Thus, primary keys for R-relations are tuples of foreign keys.

It should be clear that there is an intimate connection between the kLog data model and the E/R data model. The main difference is perhaps the restriction that primary keys must be tuples of identifiers. Whereas property values can be presented to a learning algorithm, identifiers are taken to contain no useful information outside their primary function of identifying their entity[3].

While the above assumptions put some limitations on the expressiveness of the language, kLog can effectively represent a wide family of interesting relational learning problems.

4.2.1 Illustration

To illustrate the above framework, consider the E/R diagram shown in Figure 4.1 on the left. The E/R diagram can be transformed straightforwardly in the following type signature:

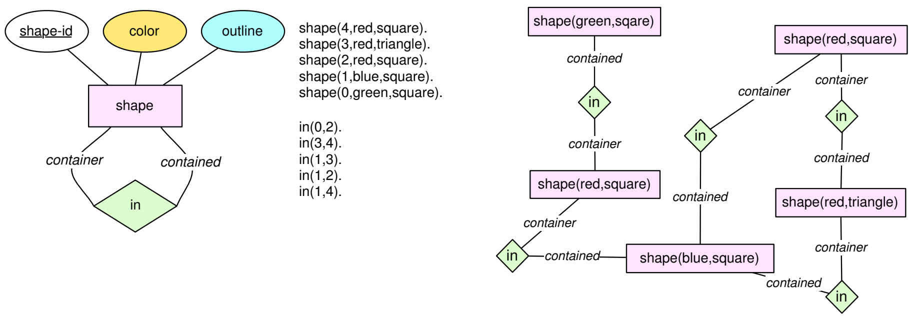
```
shape(shape-id::self, color::property, outline::property)
in(container::shape, contained::shape)
```

The relation in is an R-relation, while shape is an E-relation. The type self refers to the fact that the first argument of the relation shape contains the primary key of the relation. Furthermore, all references to the type shape will refer to that column. The primary key of in consists of the columns container

[1]This kind of relations is useful to represent *global* properties of an interpretation.

[2]Note that the word "relationship" specifically refers to the association among entities while "relation" refers the more general association among entities and properties.

[3]To illustrate the concept, it makes no sense to predict whether a patient suffers from a certain disease based on her social security number.



```
shape(4,red,square).
shape(3,red,triangle).
shape(2,red,square).
shape(1,blue,square).
shape(0,green,square).

in(0,2).
in(3,4).
in(1,3).
in(1,2).
in(1,4).
```

Figure 4.1: Graphicalization in the Bongard domain. Left: E/R diagram. Center: one interpretation z. Right: the corresponding G_z.

and contained. The roles of the columns appear in the E/R diagram. They will be important when graphicalizing the interpretation. As one can already see in Figure 4.1 on the right, the roles are used to label edges in the graphical representation of the interpretation. The middle part of Figure 4.1 contains one possible interpretation using this E/R diagram or signature. It contains ground atoms (or tuples) such as shape(0,green,square), where shape is the relation, 0 an identifier and green and square two property values.

The use of the interpretations for graphicalization will also enable the representation of symmetric relationships such as bond(atomid::atom,atomid::atom) in kLog by using the role of a column more than once. In this way, each bond can be represented by one tuple only, while a more traditional relational representation, which is directional, would require two tuples.

4.2.2 Extensional and intensional relations

An interpretation is a set of tuples or atoms, but as in deductive databases these tuples can be either listed explicitly, or they can be deduced using rules. In kLog this is realized by allowing the user to specify relations as *extensional*, in which case the tuples have to be listed explicitly, or *intensional*, in which case they are defined through Prolog clauses[4].

In the previous example, we can now add the intensional predicate inside (with type signature inside(container::shape,contained::shape) using the following definite clauses:

[4]The programmer can choose to restrict herself to Datalog notation, in which case termination can be guaranteed.

```
inside(X,Y) :- in(X,Y)
inside(X,Y) :- in(X,Z), inside(Z,Y)
```

The first clause states that whenever Y is in X, it is also inside X. The second states that whenever there exists a Z such that Z is in X and Y is inside Z that also Y is inside X. Thus these clauses define inside as the transitive closure of in.

The ability to specify intensional predicates through clauses is most useful for introducing background knowledge in the learning process. As explained in Section 4.3.2, features for the learning process are derived from a graph whose vertices are ground facts in the database; hence the ability of declaring rules that specify relations directly translates into the ability of designing and maintaining features in a declarative fashion. This is one key characteristic of kLog and, in our opinion, one of the key reasons behind the success of related systems like Markov logic.

4.2.3 Modeling small molecules

The kLog setting encompasses a relatively large ensemble of machine learning scenarios. The simplest one is classification of independent interpretations. The problem of small molecule classification as studied in the previous chapters is covered by this setting. It was pioneered in the relational learning setting in (Srinivasan et al. 1994). Each molecule is an instance. There is one E-relation, called atom (with properties such as element and charge), one relationship of relational arity 2, called bond (with a bond_type property to distinguish among single, double, triple, and resonant chemical bonds), and a zero-arity relationship called mutagenic distinguishing between positive and negative instances. A second E-relation fgroup can be used for representing moieties as in Chapter 2, plus R-relations fgmember, fg_fused, fg_connected, and fg_linked for storing which atoms belong to which moiety, and how moieties are interconnected. Figure 4.2 shows the domain representation in kLog that is (almost) equivalent to augmented molecular graphs, assuming interpretations in DMax Chemistry Assistant format[5]. The main remaining difference is that an edge in an original augmented graph representation corresponds to a vertex plus two edges in a kLog graph, as illustrated in Figure 4.3. Distances are therefore twice as large in the kLog graphs. kLog can be configured to operate as an SVM in the NSPDK

[5]The DMax Chemistry Assistant knowledge base needs to be reformatted slightly to satisfy Assumption A.1 that all entity identifiers are unique over the domain rather than over just a primary key column. To ensure this, it is sufficient to prepend an entity-set identifier to each entity identifier. Interpretation demarcation syntax also differs between the two systems.

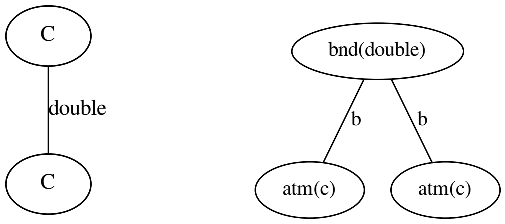
feature space, thereby, for this data representation, essentially implementing the model presented in Chapter 3.

```
begin_domain.
signature atm(atom_id::self, element::property)::intensional.
atm(Atom, Element) :- a(Atom,Element), \+(Element=h).

signature bnd(atom_1@b::atm, atom_1@b::atm, type::property)::intensional.
bnd(Atom1,Atom2,Type) :-
      b(Atom1,Atom2,NType), describeBondType(NType,Type),
      atm(Atom1,_), atm(Atom2,_).

signature fgroup(fgroup_id::self, group_type::property)::intensional.
fgroup(Fg,Type) :- sub(Fg,Type,_).

signature fgmember(fg::fgroup, atom::atm)::intensional.
fgmember(Fg,Atom):- subat(Fg,Atom,_), atm(Atom,_).

signature fg_fused(fg1@nil::fgroup, fg2@nil::fgroup)::intensional.
fg_fused(Fg1,Fg2):- fus(Fg1,Fg2,_AtomList).

signature fg_connected(fg1@nil::fgroup, fg2@nil::fgroup,
                              bondType::property)::intensional.
fg_connected(Fg1,Fg2,BondType):-
      con(Fg1,Fg2,Type,_AtomList),describeBondType(Type,BondType).

signature fg_linked(fg::fgroup, alichain::fgroup, saturation::property)::intensional.
fg_linked(FG,AliChain,Sat) :-
      linked(AliChain,Links,_BranchesEnds,Saturation),
      ( Saturation = saturated -> Sat = saturated ; Sat = unsaturated ),
      member(link(FG,_A1,_A2),Links).

signature mutagenic::extensional.
end_domain.
```

Figure 4.2: Domain specification for augmented small molecules.

4.3 Graphicalization and feature generation

The goal is to map an interpretation $z = (x, y)$ into a feature vector $\phi(z) \in \mathcal{F}$. This enables the application of several supervised learning algorithms that construct linear functions in the feature space $\mathcal{F}$. Features can be either computed explicitly or exploited implicitly, via a kernel function $K(z, z') = \langle \phi(z), \phi(z') \rangle$. Kernel-based solutions are very popular, sometimes allow faster computation, and allow infinite-dimensional feature spaces. On the other hand, explicit feature map construction may offer advantages in our setting, in



(a) Atom-bond. (b) kLog (functional group omitted).

Figure 4.3: Representation of unaugmented ethene.

particular when dealing with large scale learning problems (many interpretations) and structured output tasks (exponentially many possible predictions). Our framework is based on two steps: first an interpretation z is mapped into an undirected labeled graph G_z; then a feature vector $\phi(z)$ is extracted from G_z, or alternatively a kernel function on pairs of graphs is computed.

There are several motivations that justify this intermediate graphicalization step. First, and perhaps most importantly, graphicalization is a novel technique that is related to propositionalization, a well-known technique in logical and relational learning to transform a relational representation into a propositional one[6]. The motivation for propositionalization is that one transforms a rich and structured (relational) representation into a simpler and flat representation in order to be able to apply a learning algorithm that works with the simple representation. To the best of our knowledge, current propositionalization techniques typically transform graph-based or relational data into an attribute-value learning format, or possibly into a multi-instance learning one[7], but not into a graph-based one. The graphicalization approach that we introduce does not transform the data into an attribute-value form but rather into a graph-based format. This enables us to apply graph kernels, and in this way upgrade these kernels to accept relational input. Second, there is an extensive literature on graph kernels and virtually all existing solutions can be plugged into the learning from interpretation setting with minimal effort. This includes implementation issues but also the ability to reuse existing theoretical analyses. Third, it is notationally simpler to describe kernels and feature vectors defined on graphs, than to describe the equivalent counterpart using the Datalog notation.

[6]For a brief introduction to propositionalization, see (De Raedt 2008b, p.106–109).

[7]In multi-instance learning (Dietterich and Flann 1997) the examples are sets of attribute-value tuples or sets of feature vectors.

4.3.1 Graphicalization

The graphicalization process can be interpreted as the *unfolding* of an E/R diagram over the data. The E/R diagram is a template that is expanded according to the given groundings (see Figure 4.1). Given an interpretation z, we construct a bipartite graph $G_z = (V_z, E_z)$ as follows.

vertices $V_z = z$, i.e. there is a vertex for each ground atom (database tuple). Vertices are labeled by the predicate name of the ground atom followed by the list of property values (identifiers do not appear in the graph labels).

edges $uv \in E_z$ if $u, v \in z$, u is an E-tuple, v an R-tuple, and the primary key of u occurs as a foreign key in v. The edge is labeled by the *role* of u in v.

Labels are tuples rather than single elements as in Definition 1.2.14.

4.3.2 Graph kernel

Learning in kLog is performed using a suitable graph kernel on the graphicalized instances. While in principle any graph kernel can be employed, there are several requirements that the chosen kernel has to meet in practice. On the one hand, the kernel has to allow fast computations, especially with respect to the graph size, as the grounding phase in the graphicalization procedure can in principle generate very large graphs. On the other hand, we need a general purpose kernel with a parameterized bias to ease the injection of domain knowledge by the user.

In the current implementation, kLog uses an extension of the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK, Chapter 3). While the original kernel is suitable for sparse graphs with discrete vertex and edge labels, in the following we propose an extension to deal with dense graphs and with graphs whose labels are tuples of mixed discrete and numerical types, as occur in graphs that result when we apply the graphicalization procedure to relational datasets.

Domain knowledge bias via kernel points and viewpoints

At times it is convenient, for efficiency reasons or to inject domain knowledge into the kernel, to winnow the neighborhood subgraphs. We provide two ways to do so, via the notion of *kernel points* and *viewpoints*. These are sets of vertices induced by a user defined binary partition function $P^K(G)$ ($P^V(G)$ respectively) over the vertex set of G. We indicate that a vertex v is a kernel point with

$v \in P^K(G)$, or $v \in P^V(G)$ for a viewpoint. The neighborhood subgraphs are limited to be those rooted in P^K and P^V.

$$\kappa_{r,d}(G,G') = \sum_{\substack{\{A_v,B_u \mid R_{r,d}(A_v,B_u,G) \wedge v \in P^V(G) \wedge v,u \in P^K(G)\} \\ \{A'_{v'},B'_{u'} \mid R_{r,d}(A'_{v'},B'_{u'},G') \wedge v' \in P^V(G') \wedge v',u' \in P^K(G')\}}} \delta(A_v, A'_{v'})\delta(B_u, B'_{u'}) \tag{4.3.1}$$

Kernel points are typically vertices of some specific semantic types that are believed to represent information of high importance for the task at hand. Vertices that are not kernel points do contribute to the kernel computation but only by their presence in the neighborhood subgraphs of kernel points. In Figure 4.2, the predicate neighborhood marks relations which produce kernel points.

Viewpoints allow a finer selection and restrict only one of the elements in the pair A_v, B_u. Viewpoints are used to generate features for a specific example when a single interpretation contains multiple learning examples.

Dealing with dense graphs

The idea of counting exact neighborhood subgraph matches to express graph similarity is adequate when the graphs are sparse, that is, when the edge and the vertex set sizes are of the same order. When graphs are dense, vertices exhibit large degrees and the likelihood that any two neighborhoods with a nontrivial radius match in an exact way is almost always zero. The similarity notion then becomes degenerate. In these cases a better solution is to allow for a soft type of match between subgraphs. Although there exist several graph kernels that allow this type of match, they generally suffer from very high computational costs (Vishwanathan et al. 2010). Instead we resort to an idea introduced in the WDK (see Section 2.3.3): we consider the multinomial distribution (i.e. the histogram) of the vertex labels in the subgraph part. When dealing with dense graphs the exact match between the two pairs of neighborhood subgraphs $\delta(A_v, A'_{v'})\delta(B_u, B'_{u'})$ can be replaced by the following soft match[8]:

$$\kappa\left((A_v, B_u), (A'_{v'}, B'_{u'})\right) = \sum_{\substack{i \in V_{A_v} \cup V_{B_u} \\ i' \in V_{A'_{v'}} \cup V_{B'_{u'}}}} \delta(\lambda(i), \lambda(i')) \tag{4.3.2}$$

where $\lambda(v)$ is the label of vertex v.

[8]Note that the pair of neighborhood subgraphs are considered jointly, i.e. the labels are extracted from all vertices belonging to either of the subgraphs in the pair.

Generalized labels

A standard assumption for graph kernels is that vertex and edge labels are elements of a discrete domain. However, in kLog the information associated with vertices is comprised of a predicate name and the list of property values and can be more conveniently represented as a generic tuple containing mixed discrete and real values. In order to extend the kernel to deal with this data type we define a projector operator $\lambda_t(v)$ that returns the t-th component of the tuple associated to vector v. In addition we indicate with λ_p the projector operator associated to the predicate name. We can now extend the hard match and the soft match kernel definitions and introduce a novel kernel type.

The hard match kernel is upgraded to deal with tuple labels simply by redefining the label function as $\lambda(v) = \text{concat}\left(\lambda_p(v), \lambda_1(v), \lambda_2(v), \cdots, \lambda_t(v)\right)$. In practice we encode the whole tuple as a single element of a discrete domain.

When upgraded to deal with tuple labels, the soft match kernel becomes:

$$\kappa\left((A_v, B_u), (A'_{v'}, B'_{u'})\right) = \sum_{\substack{i \in V_{A_v} \cup V_{B_u} \\ i' \in V_{A'_{v'}} \cup V_{B'_{u'}}}} \left(\delta(\lambda_p(i), \lambda_p(i')) \sum_t \delta(\lambda_t(i), \lambda_t(i')) \right). \tag{4.3.3}$$

Thus, two subgraph pairs are compared by the overlap of their histograms for each property column.

An intermediate type of kernel can be devised by combining a variant of the hard (κ_H) and the soft (κ_S) matching notions in the product $\kappa = \kappa_H \cdot \kappa_S$. Let $\lambda = \lambda_p$ and let $\kappa_H\left((A_v, B_u), (A'_{v'}, B'_{u'})\right) = \delta(A_v, A_{v'})\delta(B_u, B_{u'})$ i.e. κ_H is the hard match kernel when we consider as vertex label the predicate name. Let κ_S be a kernel where the histograms for each property are matched in a soft way provided that the vertex has identical encoding as computed in Section 3.3 by λ^v:

$$\kappa_S\left((A_v, B_u), (A'_{v'}, B'_{u'})\right) = \sum_{\substack{i \in V_{A_v} \cup V_{B_u} \\ i' \in V_{A'_{v'}} \cup V_{B'_{u'}}}} \left(\delta(\lambda^v(i), \lambda^v(i')) \sum_t \delta(\lambda_t(i), \lambda_t(i')) \right) \tag{4.3.4}$$

in this way only the properties of vertices that correspond exactly in the two subgraphs are compared.

When dealing with real values, it suffices to substitute the exact match operator $\delta(\cdot, \cdot)$ with the product. We distinguish the projection operators λ_t for discrete properties with subscript d and those for real properties with subscript r. We

Table 4.1: One-relation tasks in kLog.

	Relational arity		
# properties	0	1	2
0	Binary classification of interpretations	Binary classification of entities	Link prediction
1	Multiclass / regression on interpretations	Multiclass / regression on entities	Attributed link prediction
>1	Multitask on interpretations	Multitask predictions on entities	Multitask attributed link prediction

can then compute the inner kernel between vertices i and i' as:

$$\kappa_{vert}(i,i') = \delta(\lambda_p(i),\lambda_p(i')) \left(\sum_d \delta\left(\lambda_d(i),\lambda_d(i')\right) + \sum_r \lambda_r(i)\lambda_r(i') \right). \quad (4.3.5)$$

4.4 Learning

kLog's representation is essentially a function $F(x,y) = w^\top \phi(x,y)$ over interpretations (x,y), where ϕ is a feature map defined by a graph kernel G and w is a weight vector. Alternatively, $F(x,y) = \sum_i \alpha_i K((x,y),(x_i,y_i))$, where the (x_i,y_i) are interpretations from the training set. This formulation assumes that the following are given:

- a set of E- and R-relations together with their signatures,
- a background theory that specifies the definition of each intensional predicate,
- a graph kernel K defining the feature map ϕ, and
- a training set of interpretations $\{(x_i,y_i)\}$.

In addition, it will be convenient to assume that the outputs y consist of a fixed set of relations $\mathcal{O}$. Depending on the type of relations in $\mathcal{O}$, one can distinguish different types of task (Table 4.1).

There are three main aspects a model in kLog needs to take care of:

1. Compute $F(x, y) = w^\top \phi(x, y)$ or, alternatively,

$$F(x, y) = \sum_i \alpha_i K((x, y), (x_i, y_i))$$

2. Maximize $F(x, y)$ with respect to y:

$$f(x) = \arg\max_{y \in \mathcal{Y}} F(x, y)$$

3. Fit w (or α) to the data.

The first step mainly involves representational issues and exploitation of background knowledge, in particular, the definition of a suitable feature space via intensional relations and via details of the kernel function.

The second step mainly involves optimization but declarative aspects of the language and background knowledge may play a role. In particular, there may be constraints that legal values of y need to satisfy.

The third step is accomplished by introducing a regularized functional via a *loss* function that measures the discrepancy between y_i and $f(x_i)$ on the training data. Depending on the choice of the loss function, the choice of the regularizer, and the choice of the optimization strategy used to minimize the regularized functional, different learning algorithms are obtained.

Actual implementations of learning algorithms in kLog are called *models*. Several alternative models are available in the current literature and many of them are pluggable into kLog. For instance, classification problems could be solved by one of many variants of SVM, as well as by logistic regression or random forests. Similarly, multi-task learning or collective classification could be handled by means of trivial algorithms or by means of algorithms capable of taking correlations and interdependencies into account. The latter are in principle more powerful but often at the expense of runtime. Most models rely on some optimization algorithm to fit the parameters to the data and again a given model could take advantage of alternative solutions (e.g. SVM could use the SMO algorithm in the dual space, or stochastic gradient descent in the primal space). Finally, kLog is able to specify supervised learning problems for which no clever (i.e. better than trivial) algorithmic solutions are known, e.g. collective regression.

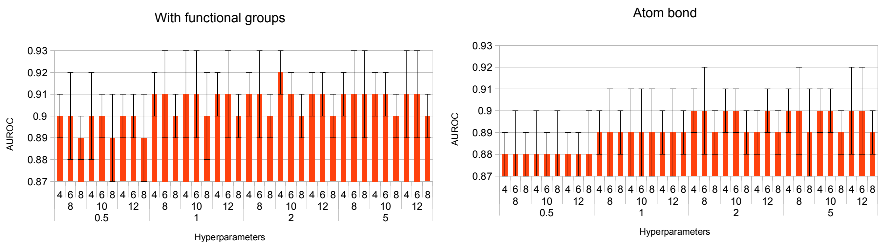


Figure 4.4: Results (AUROC) on the Bursi data set for augmented (left) and unaugmented (right) graphs. Lines on the x axis indicate kernel and SVM hyperparameters (from top to bottom: maximum radius r^*, maximum distance d^*, and regularization parameter c.

4.5 Experiments

The kLog technical report (Frasconi et al. 2011) contains more details on kLog and describes a diverse set of additional experiments (the University of Washington domain, WebKB, IMDb, and biodegradability) which we will omit in this text.

4.5.1 Bursi

An elementary kLog script is shown in Figure 4.5. The script instantiates an NSPDK feature generator with maximum radius $r^* = 4$ and maximum distance $d^* = 12$. Subsequently, tenfold cross-validation is performed of an SVM model using a polynomial kernel of degree 5 on top of the NSPDK features. With a more sophisticated script, most of the experiments of Chapter 3 can be closely approximated.

Figure 4.4 shows the results for the Bursi dataset (see Section 2.4.1). They are relatively stable with respect to the choice of kernel hyperparameters and SVM regularization and essentially match the best results reported in Section 3.7. Recall that due to the minor representational difference illustrated in Figure 4.3, distances between atoms or functional groups are twice as long in kLog graphs as in normal atom-bond graphs. Equivalent hyperparameters r^* and d^* in kLog are therefore twice as high as in Chapter 3.

```
bursi_demo(Dataset) :-
        klog_flag(klog_master,verbosity,3),
        new_feature_generator(my_fg,nspdk),
        klog_flag(my_fg,radius,4),
        klog_flag(my_fg,distance,12),
        attach(Dataset),
        new_model(model1,libsvm_c_svc),
        klog_flag(model1,c,1),
        set_klog_flag(model1,kernel_type,1),
        set_klog_flag(model1,degree,5),
        set_klog_flag(kfold_random_seed,1112),
        stratified_kfold(mutagenic, 10, model1, my_fg, muta_stratum, Performance),
        writeln(performance(Performance)).
```

Figure 4.5: kLog script to perform 10-fold stratified cross-validation for an SVM using an NSPDK feature set suitable for molecules. Domain specification (Figure 4.2) and utility predicate omitted.

4.5.2 Other applications

A second generation of kLog applications has been presented at the 2011 International Conference on Inductive Logic Programming (ILP) (Van Haaren and Van Den Broeck 2011) (Kordjamshidi et al. 2011). The challenging task in applications is to design the structure of the graph or graphs. When the domain specification is written such that neighborhoods in the graphs form coherent units of information with gradually increasing detail as the radius grows, kLog can perform well.

In national football (soccer) competitions (Van Haaren and Van Den Broeck 2011), all teams in a given league play against each other. Due to the full connnectivity, it makes little sense to represent the competition as a single interpretation (except possibly with a slicing-based approach). Neighborhoods above a certain, small radius would capture the entire interpretation. Instead, the authors chose to represent each match as an independent interpretation. Little structure then remains and the kLog-based approach essentially degenerates to a weighted form of conventional propositionalization.

A natural language processing (NLP) application (Kordjamshidi et al. 2011, Figure 1) in which the goal is to predict the presence of a word triplet that expresses a spatial relationship (such as {book, on, table}) and the location of each of its constituent words in a sentence, is a more structured domain and kLog can produce impressive results (88% precision[9] and 94% recall[10]).

The NLP application also demonstrates the usefulness of kLog as a machine learning programming language. Background knowledge such as which lexical classes are eligible to take which roles in a spatial relationship, can be easily encoded by an intentional relation. Stratification is available and the user can add custom functionality such as specific subsampling strategies in Prolog or C++.

The need to conform strictly to the E/R model is a minor challenge in some applications. In particular, the constraint that relationships are only between entities (there can be no relationships between relationships) occasionally requires the introduction of artificial entities. For the NLP application, we advised the authors to introduce candidate word function entities in addition to candidate word function relationships since spatial relationships cannot connect word function relationships directly. This workaround is always possible, but may be inelegant.

The requirement that primary keys are unique in the entire domain confuses new users and frequently requires extra effort from the user to comply with. I remain convinced that this design choice is suboptimal.

4.6 Conclusions

NSPDK (Chapter 3) has inspired kLog, a relational learning system that combines the computational efficiency of linear models with a very expressive representational language. It is moreover possible to specify a sophisticated learning bias. This is achieved through graphicalization of the data, rather than (direct) propositionalization.

The user is firmly in control during all steps of the learning process. Graphicalization is completely user-specified by means of a domain specification, which permits the definition of intensional predicates. The graph kernel can be chosen and parameterized. Finally, several learning algorithms (models) can be chosen from.

[9] Precision is the fraction of predicted spatial relationship triplets that are true.

[10] Recall is the fraction of true spatial relationships that are predicted as such, cfr. Section 2.4.2

Part II

Inverse QSAR

Overview of Part II

One aspect that is central in scientific discovery, as well as in many engineering studies, is that of determining the next experiment to be carried out. In practice, a researcher can only perform a limited number of experiments. How the experiments should be chosen depends on the goal of the research. The goal may be to obtain an accurate model of the system under study over its entire domain, using as few experiments as possible. This is the subject of the discipline of active learning. However, in many cases, large areas of the domain are of little interest, and we rather want to find the best instances (substances, compositions, designs) as evaluated on an unknown target function; again using limited experimental resources. This goal is, we believe, the more common one in chemistry, partly because chemical space is so large. It is the question that we will investigate in this second part of the thesis. We will assume that information can only be obtained by testing specific instances for their performance, whereby some cost is incurred.

In chemical terms: whereas the previous part descibed methods for the accurate and efficient prediction of the activity of given molecules, in this part, we will explore methods that deal with the inverse problem: finding molecules that exhibit a desired activity level. Effectively, input and output have been interchanged.

Chapter 5 describes this task in the context of drug discovery, in particular lead compound screening. We identify a number of suitable algorithms for the selection of screening candidates. The algorithms are compared in silico against each other and against traditional, non-incremental, high throughput screening. Chapter 5 is based primarily on (De Grave et al. 2008a).

The algorithm described in Chapter 5 is implemented and being tested in the wet lab practice by the drug discovery robot scientist project at the University of Aberystwyth. This high-profile project comprises the development of Eve, an autonomous robotic system to demonstrate the end-to-end automation of scientific discovery in the field of drug discovery. Eve is described in Chapter 6.

Chapter 7, based on (Cano Odena et al. 2008) and (Cano Odena et al. 2011), shows that the use of optimization algorithms need not be restricted to the small-molecule ligand area of the chemical space, and can just as well be applied to the other extreme of the weight scale of organic molecules: polymers. In particular, we consider the problem of the synthesis of cellulose acetate nanofiltration membranes. Nanofiltration membranes can remove low molecular weight trace contaminants in water that cannot be removed efficiently by conventional biological or physico-chemical treatments. However, membrane performance depends strongly and non-linearly on several parameters involved in membrane synthesis. It is important to find the physical parameters of the casting process that yield the best performance, as well as the optimal composition of solvents, monomer, and additives. Optimization algorithms can reduce time and material consumption to direct membrane synthesis towards better separation properties (selectivity) of the targeted compounds combined with useful fluxes.

Chapter 5

Active k-optimization for efficient ligand screening

As mentioned in the overview of Part II, an important task in many scientific and engineering disciplines is to set up experiments with the goal of finding the best instances (substances, compositions, designs) as evaluated on an unknown target function using limited resources. We study this problem using machine learning principles, and introduce the novel task of *active k-optimization.* The problem consists of approximating the k best instances with regard to an unknown function and the learner is active, that is, it can present a limited number of instances to an oracle for obtaining the target value. We use an algorithm based on Gaussian processes for tackling active k-optimization, and evaluate it on a challenging set of tasks related to structure-activity relationship prediction.

This chapter is based on (De Grave et al. 2008a) and (De Grave et al. 2008b).

5.1 Introduction

In this chapter we apply machine learning principles to select the next experiment in lead compound screening, an important step early on in the drug discovery process, in which many chemical compounds are screened against a biological assay. The goal of this step is to find a few hit compounds within the entire compound library that exhibit a very high activity in the assay. The task is akin to many other scientific and engineering disciplines, where the challenge is to identify or design those instances that have optimal performance according to some criterion that needs to be optimized. For instance:

- In coherent laser control, the goal is to find the laser pulse that maximally catalyzes a chemical reaction (Form et al. 2007).
- In nanofiltration membrane design, it is important to find those parameters of the process that yield the best permeability for water while still effectively filtering polutants, even those of very small molecular weight. This problem is discussed in detail in Chapter 7.

The common characteristic in this type of application is that the target criterion surface is unknown to the scientist or engineer, and only partial information can be obtained by testing specific instances for their performance. Such tests correspond to experiments and can be quite expensive.

In lead compound screening, it is not sufficient to find just a single optimal example. The optimal compound might ultimately not be usable as a starting point for the next step in the drug discovery process for various reasons unrelated to its performance in the assay. Therefore, a number of alternatives need to be found as well. Ideally, these near-optimal alternatives would be quite different from each other and each would have a different method of action, to increase the probability that one of them will eventually pass clinical tests. In the current chapter we will not explicitly consider this secondary concern of diversity. The challenge then is to identify the k best performing instances using as few experiments as possible. We will refer to this task as *active k-optimization.*

This task is closely related to global function optimization. It is also related to active learning in a regression setting (Cohn et al. 1996), where the goal is to find a good approximation of the unknown target function by querying for the value of as few instances as possible. Whereas this approach allows one to identify the best scoring instances, it is also bound to waste resources in the low scoring regions of the function. Thus, in contrast to active regression, an extra ingredient is added to the problem that is reminiscent of reinforcement learning. The learner will have to find the right balance between exploring the space of possible instances and exploiting those regions of the search space that are expected to yield high scores according to the current approximation of the function. Finally, active k-optimization differs also from active concept-learning as has already been applied to structure-activity relationship prediction (Warmuth et al. 2003) in that a regression task has to be performed.

This chapter is organized as follows: in Section 5.2 we formalize the problem, and in Section 5.3 we propose a Gaussian process model for tackling it. In Section 5.4 we investigate a number of different strategies for balancing exploration and exploitation. We evaluate our approach experimentally in Section 5.6. Finally, We discuss related work and possible extensions in Section 5.7.

5.2 Problem statement

Drug lead compound screening approaches typically assume the availability of a large, diverse, and fixed library of compounds. Hence, our setting is similar to the pool-based active learning setting. In this setting, the learner incurs a cost only when asking for the measurement of the target value of a particular instance, which must be selected from a known, finite pool. In principle, the learner may be able to exploit the distribution of the examples in the pool without cost. To some extent, this setting is therefore also a semi-supervised learning setting.

The problem sketched in Section 5.1 can be more formally specified as follows:

Given:

- a finite pool $\mathcal{P}$ of instances,
- an unknown function f that maps instances $x \in \mathcal{P}$ on their target values $f(x)$,
- an oracle that can be queried for the target value of any example $x \in \mathcal{P}$,
- the maximal number of queries $N_{max} < |\mathcal{P}|$ that the oracle is willing to answer,
- the number k of best scoring examples searched for.

Find:

- the top k instances in $\mathcal{P}$, that is, the k instances in $\mathcal{P}$ that have the highest values for f.

Without loss of generality, we assume that the goal is to find high, rather than low valued examples.

One can see that the above combinatorial optimization problem is a close relative to the problem of global function optimization. Algorithms developed in the discipline of global function optimization only consider $k = 1$ and are optimized for continuous domains. Still, largely the same concepts and techniques can be used.

From a machine learning perspective, the key challenge is to determine the policy for determining the next query to be asked, based on the already known examples. This policy will have to keep the right balance between exploring the whole pool of examples and exploiting those regions in the pool that look most promising.

5.3 Gaussian process model

We will use a Gaussian process model (Gibbs 1997) for learning, also known as Kriging. In this section we briefly review the necessary theory. Detailed explanations can be found in several textbooks on the subject (Rasmussen and Williams 2006, Bishop 2006). We will mostly follow (Rasmussen and Williams 2006).

We first introduce some notation. We assume that there is a feature map

$$\phi : \mathcal{P} \to F, x \to \phi(x) \tag{5.3.1}$$

mapping examples to a feature vector space F. We denote with $X_N = [x_1 x_2 \dots x_N]^\top$ the vector of the N first examples, with $T_N = [t_1 t_2 \dots t_N]^\top$ the vector of their target values, and with $\Phi_N = [\phi(x_1)\phi(x_2)\dots\phi(x_N)]$ the feature matrix, where each column is the image of an example (abusing notation in case F has infinite dimension).

For our objective criterion, $\|T_N\|_{\text{best}-k}$ is the mean[1] of the k largest elements of the vector T_N, where we assume all target values to be positive. The notation $\|\cdots\|_{\text{best}-k}$ is warranted since the function satisfies all properties of a vector norm under this assumption.

We assume that there is a linear approximate model for the target value $t(x)$ of instances

$$m(x) = w^\top \phi(x) \tag{5.3.2}$$

(with $w \in F$ a weight vector) such that the values of the modeling error $t(x) - m(x)$ for examples randomly drawn from the pool $\mathcal{P}$ are independently Gaussian distributed with zero mean and variance σ^2. We use the following notation to denote that a random variable has a Gaussian distribution:

$$P\left(t(x) - m(x)\right) = \mathcal{N}(0, \sigma^2). \tag{5.3.3}$$

or even more compactly

$$t(x) - m(x) \sim \mathcal{N}(0, \sigma^2). \tag{5.3.4}$$

[1]For our purposes, a sum would be equally suitable.

We can now compute the probability density of observing precisely T_N in the instances X_N for any given parameter vector w, a function which is known as the likelihood.

$$P(T_N|\Phi_N, w) = \prod_{i=1}^{N} P\left(T_i|\phi(x_i), w\right) \tag{5.3.5}$$

$$= \prod_{i=1}^{N} \mathcal{N}\left(w^\top \phi(x_i), \sigma^2\right) \tag{5.3.6}$$

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(t(x_i) - w^\top \phi(x_i)\right)^2}{2\sigma^2}\right) \tag{5.3.7}$$

$$= \frac{1}{\left(2\pi\sigma^2\right)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left|T_N - w^\top \Phi_N\right|^2\right) \tag{5.3.8}$$

$$= \mathcal{N}\left(w^\top \Phi_N, \sigma^2 I\right) \tag{5.3.9}$$

We use I to represent the identity matrix, $I_{ij} = \delta(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$.

Our prior belief for the vector of the coefficients w is Gaussian with zero mean and covariance matrix Σ_{pw}.

$$w \sim \mathcal{N}\left(0, \Sigma_{pw}\right) \tag{5.3.10}$$

We can compute the posterior $P(w|T_N, \Phi_N)$ from the prior and the likelihood using Bayes' rule

$$P(w|T_N, \Phi_N) = \frac{P(w|\Phi_N)P(T_N|\Phi_N, w)}{P(T_N|\Phi_N)} \tag{5.3.11}$$

$$= \frac{P(w)P(T_N|\Phi_N, w)}{P(T_N|\Phi_N)} \tag{5.3.12}$$

$$\propto P(w)P\left(T_N|\Phi_N, w\right) \tag{5.3.13}$$

$$\propto \mathcal{N}\left(0, \Sigma_{pw}\right)\ \mathcal{N}\left(w^\top \Phi_N, \sigma^2 I\right) \tag{5.3.14}$$

$$\propto \exp\left(-\frac{1}{2} w^\top \Sigma_{pw}^{-1} w\right) \exp\left(-\frac{1}{2\sigma^2}\left|T_N - w^\top \Phi_N\right|^2\right) \tag{5.3.15}$$

Adding the two exponents gives

$$w|T_N, X_N \sim \mathcal{N}\left(\bar{w}_N, \Sigma_{w,N}\right) \tag{5.3.16}$$

where the mean $\bar{w}_N$ and variance $\Sigma_{w,N}$ are

$$\bar{w}_N = \sigma^{-2}\Sigma_{w,N}\Phi_N T_N$$

$$\Sigma_{w,N} = \left(\sigma^{-2}\Phi_N\Phi_N^\top + \Sigma_{pw}^{-1}\right)^{-1}.$$

For a new example x_* we can then estimate the target value by

$$t_*|X_N, T_N, x_* \sim \mathcal{N}\left(\bar{w}_N^\top \phi(x_*), \phi(x_*)^\top \Sigma_{w,N}\phi(x_*)\right) \tag{5.3.17}$$

By defining a kernel k as

$$k(x, y) = \phi(x)^\top \Sigma_{pw}\phi(y) \tag{5.3.18}$$

and using the following abbreviations for vectors of kernel values

$$k(x_*, X_N) = [k(x_*, x_1)k(x_*, x_2)\ldots k(x_*, x_N)]^\top$$

$$k(X_N, x_*) = [k(x_*, x_1)k(x_*, x_2)\ldots k(x_*, x_N)] = k(x_*, X_N)^\top$$

$$k(X_N, X_N) = [k(x_1, X_N)k(x_2, X_N)\ldots k(x_N, X_N)]$$

for a matrix of kernel values (known as the Gram matrix), we can express the posterior distribution of t_* without explicitly refering to feature space:

$$t_*|X_N, T_N, x_* \sim \mathcal{N}\left(\bar{t}_*, var(t_*)\right) \tag{5.3.19}$$

where

$$\bar{t}_* = k(x_*, X_N)\left(k(X_N, X_N) + \sigma^2 I_N\right)^{-1} T_N \tag{5.3.20}$$

$$var(t_*) = k(x_*, x_*) - k(x_*, X_N)\left(k(X_N, X_N) + \sigma^2 I_N\right)^{-1} k(X_N, x_*) \tag{5.3.21}$$

Exact computation of the model requires a matrix inversion of cubic complexity, $O\left(N^3\right)$. There are several reduced-complexity approximations available, e.g. the Informative Vector Machine (Lawrence et al. 2003). If the examples are numerous and the feature space is small, it is more economical to compute in feature space instead (Equation 5.3.17).

5.4 Selection strategies

Different example selection strategies exist. In geostatistics, they are called infill sampling criteria (Watson and Barnes 1995, Sasena 2002). In this chapter we do not consider repeated measurements, unlike in reinforcement learning where actions can be reconsidered.

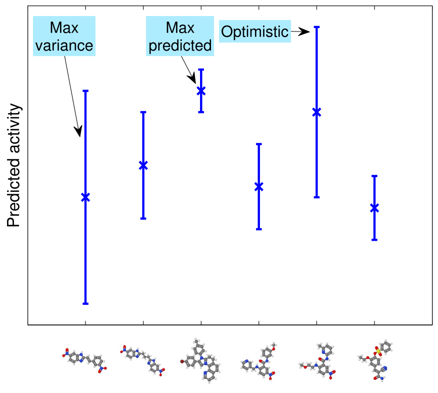
Maximum variance In active learning, in line with the customary goal of inducing a model with maximal accuracy on future examples, most approaches involve a strategy aiming at greedily improving the quality of the model in regions of the example space where its quality is lowest. One can select new examples for which the predictions of the model are least certain or most ambiguous. Depending on the learning algorithm, this translates to near decision boundary selection, ensemble entropy reduction, version space shrinking, and others. In the Gaussian process model, it translates to *maximum variance* on the predicted value or $\arg\max(var(t_*))$.

Maximum predicted Since our goal is not model accuracy but finding good instances, a more appropriate strategy is to select the example that the current model predicts to have the best target value, or $\arg\max(\bar{t}_*)$. We will refer to this as the *maximum predicted* strategy. For continuous domains, this strategy is not guaranteed to find the global optimum, or even a local optimum (Jones 2001). It is easy to see why: the model is never refined in areas where sparse initial data tentatively indicates low values, but where the lack of available data may hide high values.

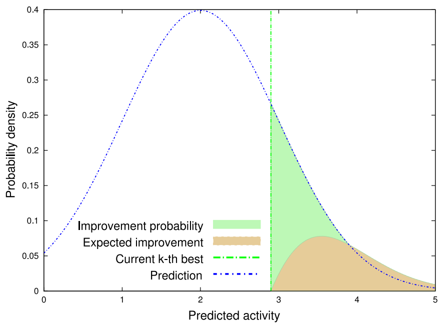
Optimism A less vulnerable strategy is Cox and John's lower confidence bound criterion (Cox and John 1997), which we will refer to as the *optimistic* strategy. The idea is not to sample the example in the database where the expected reward $\bar{t}_*$ is maximal, but the example where $\bar{t}_* + b \cdot \sqrt{var(t_*)}$ is maximal. The parameter b is the level of optimism. It determines the balance between exploitation and exploration. It is obvious that the maximum predicted and maximum variance strategies are special cases of the optimistic strategy, with $b = 0$ and $b = \infty$ respectively.

In a continuous domain, this strategy is not guaranteed to find the global optimum because its sampling is not dense (Jones 2001, pages 362-363). Informally, dense sampling means that the method will eventually sample from any given interval, no matter how small, if enough experiments are performed. It must do so independent of the target values T. A criterion that doesn't perform dense sampling, may miss a global optimum that is "hidden" in an inconspicuous interval. However, in the setting that we are considering, the pool $\mathcal{P}$ is finite, which makes this theoretical flaw less relevant.

Figure 5.1a illustrates the three strategies introduced so far. The remaining two strategies are illustrated in Figure 5.1b.



(a)



(b)

Figure 5.1: Optimization strategies for a learned model with uncertainty estimates.

Most probable improvement Another strategy is to select the example x_{N+1} that has the highest probability of improving the current solution, an idea first proposed by (Kushner 1964) and much later adapted by (Mockus 1989) and others. One can estimate this probability as follows. Let the current step be N, the value of the set of k best examples be $\|T_N\|_{\text{best}-k}$ and the k-th best example be $x_{\#(k,N)}$ with target value $t_{\#(k,N)}$. When we query example x_{N+1}, either t_{N+1} is smaller than or equal to $t_{\#(k,N)}$, or t_{N+1} is greater. In the first case, our set of k best examples does not change, and $\|T_{N+1}\|_{\text{best}-k} = \|T_N\|_{\text{best}-k}$. In the latter case, x_{N+1} will replace the k-th best example in the set and the solution will improve. Therefore, this strategy selects the example x_{N+1} that maximizes $P\left(t_{N+1} > t_{\#(k,N)}\right)$. We can evaluate this probability by computing the cumulative Gaussian

$$P\left(t_{N+1} > t_{\#(k,N)}\right) = \int_{t_{\#(k,N)}}^{\infty} \mathcal{N}\left(\bar{t_*}, var(t_*)\right) dt \ , \tag{5.4.1}$$

where $\bar{t}_{N+1}$ and $\mathrm{var}(t_{N+1})$ can be obtained from Equations 5.3.20 and 5.3.21. In agreement with (Lizotte et al. 2007), we call this the *most probable improvement* (MPI) strategy.

Maximum expected improvement Yet another variant is the strategy used in the Efficient Global Optimization (EGO) algorithm (Jones and Schonlau 1998). EGO selects the example it expects to improve most upon the current best, i.e the one with highest

$$\mathbb{E}\left[\max\left(0, t - t_{\#(k,N)}\right)\right] = \int_{t_{\#(k,N)}}^{\infty} \left(t - t_{\#(k,N)}\right) \cdot \mathcal{N}\left(\bar{t_*}, var(t_*)\right) dt \ . \quad (5.4.2)$$

This criterion is called *maximum expected improvement* (MEI).

5.5 Stopping criterion

In real-world applications it is not only important to find a solution quickly, but also to know when the optimal (or an adequate) solution has been found. The trade-off one has to make here is between budget and quality.

In a large number of situations, one will have a fixed budget and the goal will be to have an optimal solution when the budget is exhausted. Sometimes however, one can save significantly on the budget when a slightly suboptimal solution is acceptable or when the risk of having a suboptimal solution is small.

One approach is to bound the probability that any of the non-queried examples is better than the k-th best example so far. From Equation 5.4.1 we can compute for a particular example x that has not been queried the probability that its target value t will be larger than $t_{\#(k,N)}$. We can then write

$$P\left(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)}\right) \leqslant \sum_{x \in \mathcal{P} \setminus X_N} P\left(f(x) > t_{\#(k,N)}\right) \quad (5.5.1)$$

which is a tight upper bound if the individual probabilities $P\left(f(x) > t_{\#(k,N)}\right)$ are small (as is the case when we consider to stop querying) and independent.

5.6 Experimental Evaluation

As sketched in the introduction, we shall experimentally evaluate our collection of methods in the area of high throughput screening in the context of drug lead discovery. In particular, we shall evaluate the algorithms on the US National Cancer Institute (NCI) 60 anticancer drug screen (NCI60) dataset (Shoemaker 2006). In this chapter, we will not use the discretised subset of (Swamidass et al. 2005) that was described in Section 2.4.1, as we did throughout Part I, but we will rather turn to the original screening data because the model described in this chapter is capable of using, and in fact presumes, real-valued training data.

The NCI DTP repository contains measurements of the inhibitory power of tens of thousands of chemical compounds against 59 different[2] cancer cell lines. NCI reports the log-concentration required for 50% cancer cell growth inhibition (GI_{50}) as well as cytostatic and cytotoxic effect measures, but we only used the log GI_{50} data. Real world drug discovery screening operations often include non-toxicity in the measure to optimize for. For example, one could optimize for a large specificity index, usually defined as the log-ratio of the concentration of 50% toxicity for healthy cells to the concentration of 50% effectivity (in this case, growth inhibition of the cancer cells).

To perform a measurement, each compound is diluted repeatedly, yielding a geometric series of concentrations. The GI_{50} is then interpolated between the log-concentrations just above and below 50% inhibition, or obtained via a more sophisticated growth curve fit. The actual GI_{50} can turn out to be outside the range of concentrations chosen a-priori. In that case, one only knows an upper or lower bound for the value, and a new measurement for that compound must be performed to collapse the interval to a point value. We ignored such out of bounds measurements.

Repeatedly fitting a Gaussian process model in feature space would result in $O\left(N_{max}^3 \cdot |\mathcal{P}|\right)$ complexity for exact computations[3]. To avoid this, we used a linear kernel $k(x, y) = \phi(x)^\top \phi(y)$ (implying $\Sigma_{pw} = I$) with a limited number of features and performed all computations in input space (Equation 5.3.17).

The chemical structure of each compound was represented as 1024 FP2 fingerprints[4], calculated using Open Babel 2.1.0 (Guha et al. 2007). FP2 fingerprints represent the presence or absence of paths of up to seven atoms. All features are therefore binary, whereas our model allows (and assumes) real-valued features, but this proved no obstacle. The fingerprints are highly similar to the commercial de facto standard Daylight fingerprints when the Daylight Toolkit is configured to use up to seven atoms (up to eight atoms being the

[2]One of the originally 60 cell lines was evicted because it was essentially a replicate of another (Nishizuka et al. 2003).

[3]Recall from Section 5.2 that N_{max} is the experimental budget and $|\mathcal{P}|$ is the number of compounds in the pool.

[4]Open Babel never actually uses the last three bits. Depending on the dataset, other bits may also be unused.

default). We now know that this descriptor set is weak (McGaughey et al. 2007) and future research is recommended to try longer paths or altogether different descriptors (such as NSPDK, see Chapter 3, or McGaughey et al.'s TOPOSIM). A larger descriptor space, however, may require hashing to an acceptably small space or computing in input space (Equation 5.3.19) using an approximate Gaussian process model.

To improve interpretability of the experiments, an equally sized pool of 2,000 compounds was randomly selected from each assay. This also saved computational resources, though it would have been possible to use the entire dataset, for all algorithms are only of complexity $O\left(N_{max}^2 \cdot |\mathcal{P}|\right)$ as long as the number of features $dim(F)$ is constant.

The algorithms were bootstrapped with GI_{50} measurements of ten random compounds. Since the result depends on this random boot sample, each experiment was repeated 20 times and the results were averaged.

In each assay, NCI measured some compounds repeatedly. For these compounds, the dataset lists the standard deviation among the measurements, as well as the average. In order to estimate the measurement error for each assay, we used the unweighted average standard deviation over all repeated measurements in the assay. This value was used as the standard deviation σ in the Gaussian term of our model in Equation 5.3.4.

To evaluate our algorithms in practice, we recorded $||T_N||_{\text{best}-k}$ as a function of the fraction of compounds tested. For every setting (selection strategy, value of k), these functions were then averaged over the 59 datasets considered. Figure 5.2 plots these curves for $k \in \{1, 10, 25, 100\}$ for all described strategies and random selection. For the optimistic strategy, we tested optimism levels of 0.5, 1, and 2.

Table 5.1 lists for several budgets N_{max} which strategy is best (attains the highest $||T_{N_{max}}||_{\text{best}-k}$). The budget is shown as a percentage of the pool size. For each different strategy, the table then also gives the Wilcoxon signed-rank test p-value for the null hypothesis that the difference between the top-k values of this strategy and those of the best strategy is on average 0.

We are now well equipped to answer four important questions about our algorithm:

Q1 Do active k-optimization strategies isolate valuable instances quicker than random selection?

Q2 What is the relative performance of the different selection strategies listed in Section 5.4?

Q3 Do strategies that take k into account perform better than strategies that do not?

Q4 Can the stopping criterion (Equation 5.5.1) be used to decide when a near-optimal solution has been found?

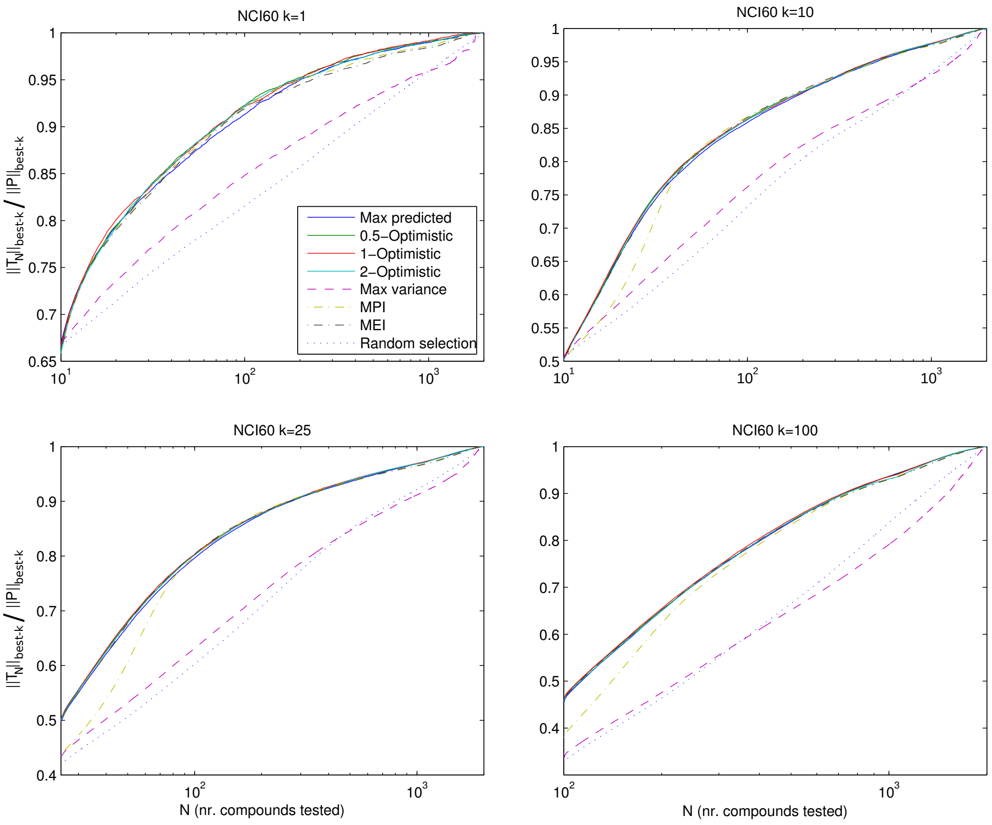


Figure 5.2: The value of $\|T_N\|_{\text{best}-k}$ in each step, for all proposed active learning strategies and random selection, averaged over 20 runs for each of the 59 datasets. A log scale is used on the horizontal axis to reveal the performance for small as well as large budgets. The vertical axis is scaled to place the aggregate target value of the overall k best compounds at one and the worst k compounds at zero.

Budget	10%	15%	20%	25%	10%	15%	20%	25%
k	1				10			
Max predicted	0.305	0.304	0.039	0.088	0.106	0.497	0.040	0.021
Optimistic b=0.5	**Best**	**Best**	0.282	0.392	0.274	0.456	0.251	0.111
Optimistic b=1	0.837	0.776	**Best**	**Best**	0.141	0.390	**Best**	**Best**
Optimistic b=2	0.898	0.472	0.094	0.229	0.179	0.298	0.179	0.298
Max variance	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
MPI	0.946	0.538	0.108	0.174	0.455	0.230	0.189	0.052
MEI	0.037	0.057	0.005	0.047	**Best**	**Best**	0.934	0.809
Random	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
Budget	10%	15%	20%	25%	10%	15%	20%	25%
k	25				100			
Max predicted	0.074	0.437	0.015	0.015	0.046	0.063	0.003	0.010
Optimistic b=0.5	0.202	0.319	0.177	0.192	0.197	0.118	0.007	0.022
Optimistic b=1	0.280	0.634	**Best**	**Best**	**Best**	**Best**	**Best**	**Best**
Optimistic b=2	0.083	0.673	0.264	0.478	0.042	0.170	0.016	0.068
Max variance	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
MPI	**Best**	**Best**	0.385	0.141	10^{-5}	0.005	0.003	0.001
MEI	0.487	0.184	0.158	0.083	0.254	0.492	0.019	0.001
Random	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

Table 5.1: Performance of active k-optimization strategies. The strategy that achieved highest $\|T_N\|_{\text{best}-k}$ after selecting the indicated percentage of all compounds is marked **Best**. For the other strategies, Wilcoxon signed-rank p-values are shown, where a p-value below a given significance level indicates that the strategy is inferior. ϵ indicates that $p < 10^{-8}$.

5.6.1 Expedience

From the results presented in Table 5.1 and Figure 5.2, one can see that random example selection clearly performs worse than all other selection methods in all settings, except for the maximum variance strategy which does still worse for large budgets, especially for $k = 100$. We can conclude that the answer to question **Q**1 is positive, because actively choosing examples with one of the presented strategies substantially speeds up the finding of examples with high target values.

It is remarkable that the starting points of the random strategy are lower for higher k. This is due to the fact that the distribution of target values is skewed: the lower end of the range of target values is more sparsely populated than the area of very large target values (presumably because finding a good compound encourages researchers to test similar compounds, which are likely to perform similarly). In this way, $\|T_N\|_{\text{best}-k}$ decreases only slowly while $\|T_N\|_{\text{worst}-k}$ (the average of the k smallest elements of T_N) increases quickly for larger k. This causes the value of a random sample to be lower when scaled to a $[0, 1]$ interval. That the non-random strategies start higher than the random strategy for $k = 25$ and $k = 100$ is due to the fact that there are only 10 bootstrapping examples, and the non-random strategies actively select 15 (for $k = 25$) or 90 (for $k = 100$) examples before they can be evaluated for the first time.

5.6.2 Relative performance

Unsurprisingly, one can see that querying the maximally uncertain example is (in contrast to settings where one tries to optimize accuracy) not a good k-optimization strategy.

Overall, on the NCI60 datasets, the optimistic strategy with an optimism level of 1 was most robust. In all situations considered, it performed either best or not significantly worse than the best strategy. The difference with 2 and 0.5 optimism is more consistent for higher values of k. In Figure 5.2 we can see that the relative differences between the optimistic, maximum predicted, and MEI strategies are modest. A possible contributing factor is that the data set is skewed and not the result of random screening.

Note that we exploited the information in the NCI datasets about the accuracy of the measurements. For other datasets that do not allow to estimate the noise level in the input data, it may be harder to come up with a good value for $var(t_*)$. One can use a maximum likelihood estimate, at the cost of some robustness (Sasena 2002).

Greedily querying the example for which the highest target value is predicted, performs slightly worse than the optimistic strategy. The MPI strategy performs worse than the optimistic strategies in the very beginning, except for $k = 1$. It performs (and allegedly behaves) similarly to the maximum variance strategy when it hasn't seen many more examples than the 10 random bootstraps. From about 5% for $k = 10$ and 10% for $k = 25$, its performance is competitive and sometimes best, but it again becomes suboptimal for high budgets. The MEI strategy performs extremely well for $k = 10$, but is outperformed in some other settings. This concludes our answer to question **Q**2.

5.6.3 Utility of advance knowledge of k

In Table 5.2 and Figure 5.3 we see that the active learning strategies that explicitly take k into account, perform far better than their global optimization ($k = 1$) peers, except for the warmup of MPI. However, from question **Q2** we learned that the most robust strategy on our datasets, 1-optimism, performs as well. Since optimism does not rely on prior knowledge of k, the answer to question **Q3** is negative.

Budget	10%	15%	20%	25%
MPI $k_{target} = 1$	10^{-7}	ϵ	ϵ	ϵ
MEI $k_{target} = 1$	ϵ	ϵ	ϵ	ϵ
MPI $k_{target} = 25$	**Best**	**Best**	**Best**	**Best**
MEI $k_{target} = 25$	0.487	0.184	0.394	0.640

Table 5.2: Influence of prior knowledge of k on the performance of k-dependent strategies. The k-dependent strategy finding the best $\|T_N\|_{\text{best}-25}$ is indicated ($k_{eval} = 25$). For the other strategies, we show p-values for the null hypothesis of equal performance. ϵ indicates that $p < 10^{-8}$.

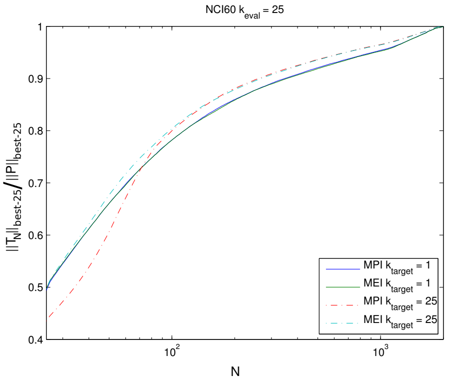


Figure 5.3: The value of $\|T_N\|_{\text{best}-25}$ in each step, for the MPI and MEI strategies, optimizing for either k=1 or k=25.

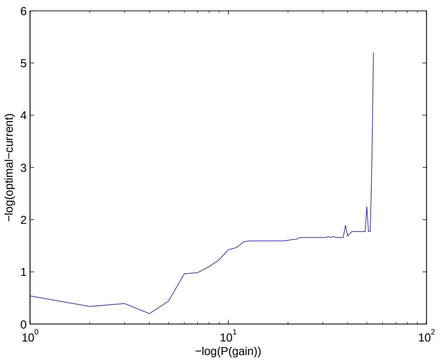


Figure 5.4: Stopping criterion: the negative logarithm of the difference between the optimal solution and current solution plotted against the negative logarithm of predicted probability of suboptimality according to Equation 5.5.1

5.6.4 Stopping criterion

To evaluate the stopping criterion, we used Equation 5.5.1 to estimate the probability $P\left(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)}\right)$ that there exists an unseen example in the pool $\mathcal{P}$ which is better than the k-th best seen so far. We did so during one experiment with the MPI strategy for every dataset, and recorded these probabilities together with the differences between the solution at that point and the optimal solution. In this way, we can evaluate how much value one would lose on average if one would stop the screening when the probability of finding anything better would drop below a certain threshold.

In Figure 5.4, the negative logarithm of the differences between solution so far and optimal solution, i.e. $-\log\left(\|f(\mathcal{P})\|_{\text{best}-k} - \|T_N\|_{\text{best}-k}\right)$, is plotted against the negative logarithm of the estimated probability that there is still a better solution, i.e. $-\log\left(P\left(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)}\right)\right)$. The standard deviations on the points in this curve are all below 0.2.

From Figure 5.4 one can see that there is a good relation between the estimated probability that the best solution has not yet been found and the optimality of the current solution. In particular, when the stopping criterion predicts a very small probability of finding a better solution, one can be confident that querying more examples will not be very useful. This answers question **Q**4 positively.

5.7 Related work and possible extensions

To summarize, we introduced the active k-optimization problem in a machine learning context, we developed an approach based on Gaussian processes to tackling it, and we applied it to a challenging structure-activity relationship prediction task, demonstrating good performance.

This chapter is related to several articles that combine kernel methods and Gaussian processes both in the machine learning and the global optimization communities. In machine learning one aims at improving prediction accuracy, and common strategies select the most uncertain examples, or select the examples that maximize information gain. In global optimization, Gaussian processes are a popular surrogate to save on expensive function evaluations (Sasena 2002, Lizotte et al. 2007). The advances in global optimization were preceded and inspired by geostatistics (Watson and Barnes 1995). One can easily recognize the MPI, MEI, and maximum variance concepts in the theoretical work of Watson and Barnes.

In the area of lead compound screening, active learning has so far only been applied for classification or regression purposes but not for optimization. E.g. (Warmuth et al. 2003) shows that the maximum-predicted strategy works well for discriminating rare active compounds from inactives using an SVM. Furthermore, the NCI database has been used as benchmark for several machine learning approaches, e.g. (Swamidass et al. 2005, Ceroni et al. 2007, Menchetti et al. 2005), and see also Chapters 2 and 3. Note that most machine learning papers use the discretised GI_{50} of a specific subset of compounds (Swamidass et al. 2005) to evaluate binary classification algorithms, whereas in this chapter we used the real GI_{50} of a different subset. As the results show, classification of compounds can be learned to a certain extent, but accurate prediction (classifying borderline cases) is still harder than finding extreme values as in our setting.

Two interesting further questions for research are

1. whether one could make further gains by devising a strategy that also takes into account a budget that is fixed from the start, and
2. whether one can select several examples to be queried together in a single batch before getting the target values for all of them. This is often needed in high throughput screening (HTS).

To address the first question, one could e.g. focus the first fraction of the budget more on exploration and the last part only on exploitation.

The second question requires one to spread selections over the space in order to avoid obtaining too many correlated values (Guestrin et al. 2005). A few authors have touched upon this problem in the context of surrogate-based optimization, but the batch size was algorithm driven as opposed to application constraint driven, e.g. (Jones 2001). This raises a more general problem: given some collected X_N, T_N training data and a pool $\mathcal{P}$ of examples that one could query next, select n new examples to query. In such a situation, it may not be optimal to select examples that individually optimize some criterion. In the ideal case, one would like to optimize the joint contribution of the entire batch. E.g. if $k = 1$, the probability that querying examples $x_{N+1} \dots x_{N+n}$ would improve the solution would be

$$P\left(\max\{t_{N+1} \dots t_{N+n}\} > t_{\#(k,N)} \mid X_N, T_N\right) \tag{5.7.1}$$

which evaluates to the integral of a Gaussian over a union of half-spaces. For large n, it is nontrivial to select the n examples for which this value is maximized. However, one can efficiently select $x_{N+1} \dots x_{N+n}$ in order, such that in every step the example x_{N+i} is selected that maximizes

$$P\left(\max\{t_{N+1} \dots t_{N+i}\} > t_{\#(k,N)} \mid X_{N+i-1}, T_{N+i-1}\right) . \tag{5.7.2}$$

The batch selection generalization of the MEI heuristic would be the $\arg\max_{x_{N+1}\dots x_{N+n}}$ of:

$$\mathbb{E}\left[\max\left(0, \left(t_{N+1} - t_{\#(k,N)}\right), \dots, \left(t_{N+n} - t_{\#(k,N)}\right)\right)\right] \tag{5.7.3}$$

$$= \int_{t_{\#(k,N)}}^{\infty} \cdots \int \left(\max\left(t_{N+1}, \dots, t_{N+n}\right) - t_{\#(k,N)}\right) \cdot \mathcal{N}^n\left(\bar{t_*}, var(t_*)\right) dx_{N+1} \dots dx_{N+n} \tag{5.7.4}$$

$$\propto \int_{t_{\#(k,N)}}^{\infty} \cdots \int \max\left(t_{N+1}, \dots, t_{N+n}\right) \cdot \mathcal{N}^n\left(\bar{t_*}, var(t_*)\right) dx_{N+1} \dots dx_{N+n} \tag{5.7.5}$$

We will reiterate this problem in the next two chapters.

Chapter 6

Eve: the drug discovery robot scientist

A robot may not injure humanity,
or, through inaction, allow
humanity to come to harm.

Zeroth Law of Robotics
Isaac Asimov

In the previous chapter, we have demonstrated that selection strategies based on a Gaussian process model can successfully select chemical compounds to be tested in an expensive assay. In this context, "success" means that relatively few experiments (compounds tested) are required to find the good compounds in the library. In this chapter, we will find out how the optimization algorithms can be implemented in a real-world laboratory.

6.1 Introduction

An important practical consideration when implementing an automatic selection procedure for experiments, is to what extent the execution of the experiments can also be automated. Since the decision making is already automated by the algorithm, the system clearly becomes more valuable when the need for human intervention for the mindless execution of the prescribed experiments is reduced. An even more desirable system would be obtained if one manages to also automate the objective interpretation of the experiments. The ultimate system would be one in which all aspects of the endeavour are fully automated. Since, in some sense, the system then performs research independent of human intervention, we may call it a scientist — a robot scientist.

More precisely, a robot scientist is a system which automates all aspects of the scientific discovery process: it generates hypotheses from a computer model of the domain, designs experiments to test these hypotheses, physically performs the experiments using robotic systems, interprets the resulting observations, updates its domain model accordingly, and repeats the cycle. The distinctive feature compared to regular lab automation systems, data analysis software, and experimental design tools, is that a robot scientist is able to autonomously execute the entire closed loop of scientific discovery. It can continuously refine its model of the world through rational experimentation.

Eve (Sparkes et al. 2010) is the second robot scientist developed at the lab of professor Ross King at Aberystwyth University in Wales. The earlier robot scientist is called Adam (King et al. 2009). There was also an earlier prototype which was not fully automated (King et al. 2004). Adam and Eve still need human intervention for physical work that is outside of their core experimental capabilities, such as restocking, waste disposal, and other maintenance chores. Their makers therefore consider them prototypes of an ideal, fully autonomous robot scientist.

Adam performs yeast culture growth experiments with the yeast *Saccharomyces cerevisiae*. Its goal is to find out which genes code for metabolic enzymes that are known to exist in the organism, but for which the gene is not yet identified. Adam has already discovered the genes for several such enzymes, contributing new knowledge to biology.

Whereas Adam performs pure biological research, Eve is a drug discovery researcher. It is a system to demonstrate the automation of closed-loop learning in drug screening and design, that is, demonstrate machine learning and automated quantitative structure-activity relationship (QSAR) deeply integrated into the drug screening and design processes. The main goal is to ultimately improve both the efficiency and quality, as well as reduce the cost, of drug discovery.

In this chapter, we describe an algorithm for experiment selection that we developed for Eve, starting from the active k-optimization framework exhibited in Chapter 5. We will first describe the details of Eve's design that determine which types of experiments can be performed on the robot.

6.2 Eve's anatomy

6.2.1 Hardware

Since we have not contributed to Eve's hardware design, we rely on (Sparkes et al. 2010), the documentation at the project's website[1], and personal communication with the design team for a description of Eve's design and capabilities. Some of the text is taken from (Sparkes et al. 2010).

Eve's laboratory robotic system (Figure 6.1) contains a set of instruments designed to give the system the flexibility to prepare and execute a broad variety of biological assays, including: cellular growth assays, cell based chemical compound screening assays, and cellular morphology assays. There are three types of liquid handling instruments included in the system, one of which uses advanced non-contact acoustic transference. For observation of assays, the system contains two multi-functional microplate readers capable (with the appropriate filters) of recording measurements across a broad range of both excitation and emission wavelengths. There is also an automated cellular imager capable of taking images of the well contents of microplates using both bright-field and a broad range of other wavelengths. The primary biological assays used on Eve create one or more fluorescent protein markers that can be detected on the readers and imager, such that Eve can not only quantify the amount of marker produced using the readers, but also potentially localise it to specific cellular regions or organelles using the imager. Eve also utilises control software for the robotic system that is flexible enough to allow to reconfigure the experimental process. Both hardware and software subsystems are designed to allow rapid reconfiguration to carry out a number of different biological assays. The designers claim that the system is equivalent to the best systems available in the pharmaceutical industry.

6.2.2 Compound library

Eve initially uses a robot-accessible compound library of 14,400 chemical compounds: the Maybridge HitFinder™ library. This compound library was developed specifically to contain a diverse range of compounds. It was selected as a subset of a large compound library by a two-stage filtering process based first on Lipinski's Rule of Five (Lipinski et al. 1997) to reduce the set to 200,000 compounds, then secondly by using a pharmacophore fingerprinting process (Butina 1999) and cluster analysis to further reduce the set to 14,400 compounds. HitFinder is not a large compound library by industrial standards;

[1] `http://www.aber.ac.uk/en/cs/research/cb/projects/robotscientist/`

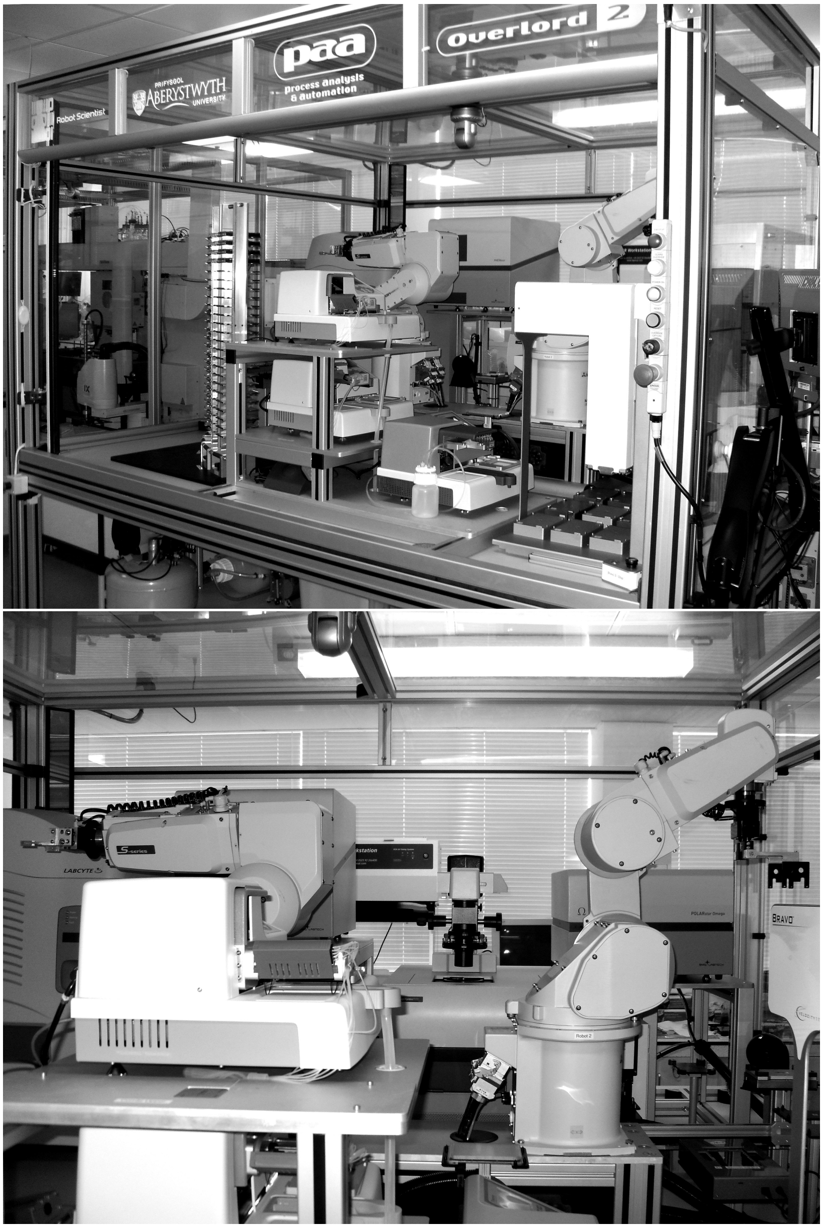

Robot Scientist
PRIFYSGOL ABERYSTWYTH UNIVERSITY
paa
process analysis & automation
Overlord 2
LABCYTE
S-series
BRAVO
Robot 2

Figure 6.1: Eve: (top) overview, (bottom) close-up of part of the machine.

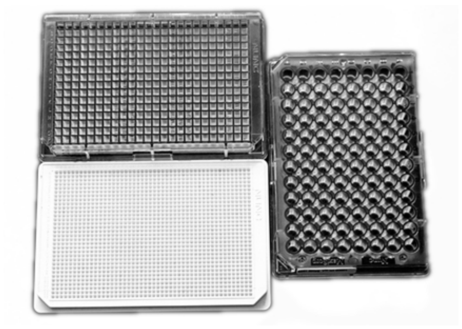


Figure 6.2: Standard microtiter plates of 96, 384, and 1536 wells. Eve uses 384-well plates with opaque walls during normal screening operation. Public domain image from NCI.

a pharmaceutical company may have many hundreds of thousands or even millions of compounds in its primary screening library. The aim of the robot scientist project is to demonstrate the proof-of-principle that incorporating machine learning and QSARs into the screening process can improve on the current mass screening approach.

6.3 Screening operation

6.3.1 High throughput screening

As any modern screening lab, Eve uses microtiter plates or microplates (Figure 6.2) to handle all liquids. Microplates are rectangular arrays of wells, each of which can contain a different experimental solution.

Eve is physically capable of screening at a moderately high throughput rate of over 10,000 compounds per day, at a single concentration and assuming no imaging is required. For the yeast growth assays it uses, throughput is primarily limited by the need to repeatedly monitor the yeast cultures as they grow. In high throughput mode, Eve can simultaneously incubate 8 microplates, containing up to 320 compounds each[2], and still observe the fluorescence of the cultures at an adequate frequency to record detailed growth curves. Three yeast strains can be observed simultaneously if they express different fluorescent

[2]Some of the 384 wells are used for validation compounds, background fluorescence measurements, and other overhead necessary to obtain reliable measurements.

colors. The yeast cultures are grown for two days, giving a throughput of 1280 compounds per day. In this mode, the compounds to be screened can only be selected at the source plate level, in predefined groups of 320 compounds.

The Eve system includes all necessary software components to allow it to perform cycles of targeted drug screening. There are three stages to this approach; mass screening, hit verification, and hypothesis-driven targeted screening. For each of these stages, experiment design software generates the biological experiment plans, which combine chimeric yeast target strains and chemical compounds.

6.3.2 Cherry picking

Eve starts a screening project in high throughput mode, continuously monitoring the results. When a sufficient number of tentative hits has been detected, it stops the mass screen and verifies the tentative hits. The microplate layout used for the verification is very different from high throughput mode, where only a single well is assigned to each compound. Each compound is tested at multiple concentrations and several redundant copies are made at each concentration, so that accurate dose-response characteristics can be observed. Currently, a series of eight dilution steps are used, comprising an equidistant series on a logarithmic concentration scale. Six replicates of each dilution series are distributed over the plate to avoid measurement errors as much as possible. Eight compounds can thus be fit on a 384-well plate. Since Eve can read the growth curves of eight simultaneously incubated plates, the optimal batch size for detailed screening is 64 compounds. This mode of operation is called cherry picking mode, because the compounds layed out on the experimental microplates can be arbitrarily picked from the library. By contrast, each high throughput screening microplate is a diluted copy of one of the mother plates in the library.

After verifying the hits, Eve then switches to a more targeted approach, using machine learning and QSARs to relate the experimental results to the chemical structures of the compounds and possibly other background knowledge of the assay. It will generate hypotheses about what it considers would be useful compounds to test next. It then plans the (cherry picking) screening experiments to test these hypotheses, performs these experiments on the robotic system, uses machine learning to analyse these results, and then iteratively cycles around testing other compounds until it can identify the best set of lead compounds for the target. Eve will first test those compounds which are available from its own compound library, then suggest other compounds that are commercially available and should be tested. Potentially, Eve could even suggest new compounds that should be synthesised for testing.

Eve's hardware assembly was completed in early 2009. At the time of writing, parts of the low-level cherry picking software are still under development, but the high throughput screening of a few assays (Section 6.7.1) had been completed.

6.4 Hit expansion

6.4.1 Comparing traditional hit expansion with support vector regression

The data from exhaustive high throughput screening of the HitFinder library for activity against *Plasmodium vivax*, a prevalent malaria parasite, was modeled by Support Vector Regression (using SVM-Light) using the NSPDK (Chapter 3) over augmented graphs (Chapter 2). The model was used to select 11 additional compounds for purchase from the 56,391 compound Maybridge Screening Collection library (the 12 highest predicted compounds except one which was unavailable). An additional 6 compounds were selected using a more conservative method: the nearest neighbors (NN) (as measured by traditional fingerprints) of a strong hit that was manually picked as the most promising avenue.

The NSPDK parameters for these predictions were set as follows: maximum radius $r^* = 2$, maximum distance $d^* = 4$, and composition with a polynomial kernel of degree 3. The hash was limited to 22 bits. The regression SVM's training error versus margin trade-off c was set to 1.

6.4.2 Results

The 11 compounds selected with the NSPDK SVR model yielded 1 strong hit and 3 weak hits. The NN approach was numerically even more succesful with 3 strong hits and 1 weak hit. Note that the abundance of strong hits in the training set was slightly below 0.1%.

The hits identified by NN are of course far more similar to the training hitlist, and hence less informative, than the hits found using SVR. The average similarity[3] of the strong NN hits to the respective nearest previously known strong hit is 0.813 (0.681 for the most dissimilar). By contrast, the strong SVR hit was very dissimilar at only 0.263.

[3]We measure this similarity using the Tanimoto kernel over Open Babel FP2 fingerprints, see Section 5.6.

6.5 Teaching Eve active k-optimization

We have adapted the Gaussian process model based algorithm for active k-optimization, decribed in Chapter 5, to make it suitable for making Eve's experiment selection decisions in cherry picking mode. The single, crucial difference of this application with the setting as exhibited in Section 5.2 is that for Eve, it is very inefficient to screen only a single compound and then wait for the results of that single compound before deciding on the next. As sketched in Section 6.3, the system uses microplates to handle liquids. Leaving wells on the microplates empty wastes capacity of the robotic system and in general reduces the ability to distribute fixed costs (such as the cost of the microplate itself) over multiple experiments. This is in fact the case for most real-world drug discovery wetlabs and the problem was already touched upon in Section 5.7.

The use of relatively slow (but reliable and information-rich) cell culture growth assays makes this limitation even more stark: each experiment takes approximately two days to grow the yeast culture, independent of the number of compounds to be tested. Therefore, any practical experiment selection algorithm for Eve must be able to select a batch of compounds in parallel. As was mentioned in Section 6.3.2, the optimal batch size for Eve is 64. This number has been determined only recently. Earlier in the design of the robotic system, optimal batch sizes of 2, 4, or 8 were projected.

While we have developed a version of the MEI strategy (see Chapter 7) capable of constructing almost-locally-optimal experiment batches, it is unfortunately computationally far too demanding for Eve's large batch size and large number of training points. Eve is a real-time environment, where the selection of experiments must happen reasonably quickly to achieve high utilisation of the expensive robotic laboratory. This would be even more important if fast protein-based assays are used rather than yeast cell growth assays, as is often the case in industry[4]. In this context, decisions must be made in seconds, rather than hours or days.

Our attempts have failed so far to dramatically optimize the computation of batch-selection MEI by analytically solving the multi-dimensional version of integral 5.4.2, which would completely eliminate the need for Monte-Carlo numerical computation.

[4]A recent paper (Swinney and Anthony 2011) reveals that phenotypic assays have yielded more FDA-approved first-in-class small molecule drug discoveries than target-based assays, even though the latter dominate R&D efforts. Excessive focus on target-based assays may contribute to high attrition rates and low R&D efficiency. Eve's yeast growth experiments occupy a middle ground, combining characteristics of both full disease phenotype reproduction and focused single-target assays.

A simple solution to select batches of experiments based on a model-based optimization heuristic that can only select one experiment at a time, is to use the algorithm iteratively, each time adding the selected experiment to the database of experiments that have already been carried out. We simply assume that the predicted quality of the compound is exactly correct. We choose to implement in this way the most robustly performing algorithm according to our in silico experiments exhibited in Section 5.6: Gaussian process model based optimization with an optimistic selection strategy. When composing a batch of compounds for experimentation, the heuristic avoids picking an additional compound that is very similar to any of the already selected compounds, because the presence of the already selected compounds in its database of training examples makes it 'believe' that the uncertainty in that area of the chemical space is low. A higher level of optimism d leads to more diversity in the batch, as well as between batches.

The algorithm is implemented as an Octave 3[5] program and makes use of Open Babel 2.2.3 (Guha et al. 2010) FP2 fingerprints and supporting code in Java and Bash. A persistent, memory-mappable compound feature matrix is kept in sync with the main relational database containing the compound structures.

6.6 Related work

The work most related to the heuristic was already mentioned in Sections 5.4 and 5.7 on active k-optimization.

Multi-pathogen assay screens require multi-objective optimization, 'multi' in practice being two or three. Eve's instruments have a technical limit of five wavelengths. Our current approach does not yet deal with multi-objective optimization. There is a significant body of literature on multi-objective optimization with evolutionary algorithms (MOEA). A recent overview of MOEAs can be found in (Knowles 2009). While these algorithms assume a continuous example space, or at least an example space equipped with operators such as mutation and crossover, several of them may be adapted to the chemical space or to operate on a finite pool of chemicals $\mathcal{P}$ if assisted by an appropriate distance function or kernel. This begs the question whether there is a MOEA algorithm that would be suitable for guiding Eve's decision making in cherry picking mode.

Some properties of MOEAs make them less suitable for implementation in Eve. Very few MOEAs are frugal in the number of evaluations (experiments)

[5]`http://www.octave.org`

required, with the exceptions (Beume et al. 2007, Knowles 2006) achieving their frugality by assistance from a machine learning model and heuristics like those exhibited in Chapter 5. According to (Coello Coello 2006, page 13), model-assisted MOEAs must necessarily be limited to problems of low dimensionality. This is in agreement with (Knowles and Hughes 2005), although the latter authors do not put it so strongly. The chemical space is definitely not of low dimension. The problems we consider, however, have only a finite number of possible solutions, so the limitiation may be circumvented.

Another issue is that MOEAs are usually tuned to discover the entire Pareto surface. By contrast, Eve should not search for trade-off compounds that moderately inhibit several pathogens but that act against none of the diseases strongly enough to be used in patients. Such compounds are only to be examined if the experiment yields actionable information for discovering a compound that is maximally effective on either of the assays.

A recent result that is particularly interesting in the context of Eve is the efficient computation of the MEI criterion (Equation 5.4.2) for two and three dimensional targets (Emmerich et al. 2008).

6.7 Cherry picking simulation

6.7.1 Data

At the time of writing, the low-level robotic control software for cherry picking was not yet robust enough to allow closed-loop experimentation with our algorithm. Fortunately, the high throughput screening mode was functional: two multi-strain assay screenings on the entire HitFinder collection had been completed. Using this first experimental data obtained by Eve, we have simulated in silico retrospectively what would happen in cherry picking mode in vitro. However, as described in Section 6.2, high throughput screening data are of lower quality and of a somewhat different nature than cherry picking data, so the results should be interpreted with care.

Both assays consisted of three transgenic yeast strains, each presenting a different foreign Dihydrofolate reductase (DHFR) enzyme. DHFR is an essential enzyme, so successful blocking of DHFR inhibits growth of the cell culture. A gene for the removal of toxins was knocked out in all strains in an attempt to improve the signal-to-noise ratio. Each of the strains also produced a fluorescent protein of a different color, enabling the microplate reader to measure the growth of the strains independently. For both assays, two of the strains featured DHFR obtained from a pathogen, and one of the strains expressed the corresponding human DHFR. The pathogens were malaria-causing *Plasmodium falciparum* (Pf) and *Plasmodium vivax* (Pv), *Trypanosoma brucei* (Tb) which causes sleeping sickness, and *Schistosoma mansoni* (Sm), the parasite causing intestinal schistosomiasis. The goal is to find a ligand that strongly inhibits at least one of the pathogen DHFRs, but not the human variant.

6.7.2 Method

We translated the goal into a set of single-objective optimization problems. We want to maximize these objective functions:

$$OF_{Pathogen} = \frac{\Delta I^{\nu(Human)}_{compound}}{\Delta I^{\nu(Human)}_{control}} - \frac{\Delta I^{\nu(Pathogen)}_{compound}}{\Delta I^{\nu(Pathogen)}_{control}} \tag{6.7.1}$$

where I is fluorescence intensity, Δ indicates that we subtract the (background) intensity at the start of the growth experiment from the final intensity from the saturated cell culture, $\nu(Human)$ and $\nu(Pathogen)$ are the respective fluorescent light colors, and *compound* or *control* indicates whether a test compound was present or not. All yeast strains used for the experiments grow substantially in control conditions and have a substantial default expression of the fluorescence gene in the control medium, so the objective poses little risk for numerical instability or division by zero.

Out of 14,400 compounds, 99 were found to be toxic to yeast, or to be autofluorescent in the spectra used for the assay. In neither case can the DHFR inhibition level be measured. We excluded these compounds from the dataset. Since both properties are independent from the target protein under investigation, it is sensible to also exclude the compounds from future screens.

The efficacy of the optimization procedure was tested on the mass screening data a posteriori, starting from a small number of randomly screened compounds. Average scores and standard deviation over many random bootstraps were collected.

6.7.3 Evaluation method

As in the previous chapter, we compare the measure $\|OF\|_{\text{best}-k}$ with random screening. We also evaluate the optimization in terms of how many compounds are superior to the best compound discovered, but go undetected due to not screening the entire pool.

How many superior compounds are left undetected when randomly screening a part of the pool? Assume for mathematical convenience that no two compounds in the pool have exactly the same objective function value. The compounds can then be strictly ordered by increasing value and we assign a sequence number to each compound: $1, 2, \ldots, |\mathcal{P}|$ where $|\mathcal{P}|$ is the size of the compound pool $\mathcal{P}$ as in the previous chapter. Random selection screening with a budget for testing M compounds can be interpreted as selecting without replacement M numbers from the set $\{1, 2, \ldots, |\mathcal{P}|\}$, following a uniform distribution. To answer the question of the number of undetected superior compounds, we can compute the expected maximum number in a sample of size M and subtract it from $|\mathcal{P}|$.

The total number of possibilities to choose M numbers from $\mathcal{P}$ is $\binom{|\mathcal{P}|}{M}$. The number of possibilities to select M numbers of which the maximum is k, is[6] $\binom{k-1}{M-1}$. The expected maximum of the selection is thus

$$\frac{\sum_{k=M}^{|\mathcal{P}|} k\binom{k-1}{M-1}}{\binom{|\mathcal{P}|}{M}} = \frac{M}{\binom{|\mathcal{P}|}{M}} \sum_{k=M}^{|\mathcal{P}|} \frac{k}{M}\binom{k-1}{M-1} \tag{6.7.2}$$

$$= \frac{M}{\binom{|\mathcal{P}|}{M}} \sum_{k=M}^{|\mathcal{P}|} \binom{k}{M} \tag{6.7.3}$$

$$= \frac{M}{\binom{|\mathcal{P}|}{M}} \binom{|\mathcal{P}|+1}{M+1} \tag{6.7.4}$$

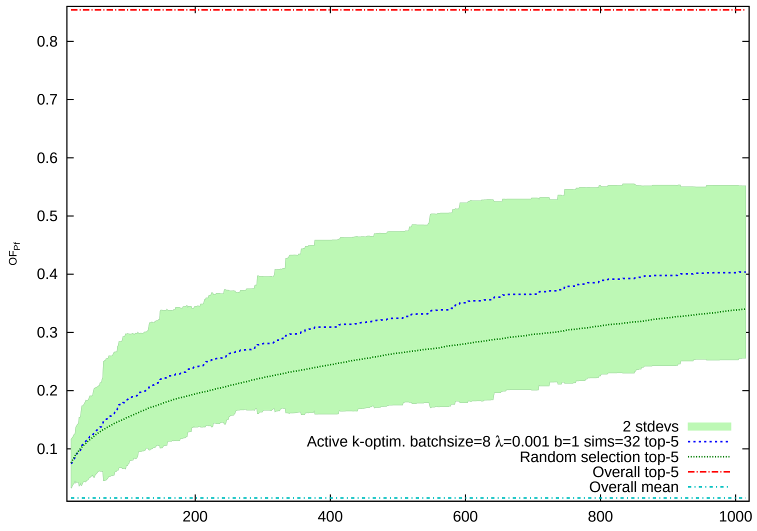
$$= \frac{M(|\mathcal{P}|+1)}{M+1} \tag{6.7.5}$$

The equality of Equations 6.7.3 and 6.7.4 can be obtained by induction on Pascal's rule $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$. We subtract the expected maximum 6.7.5 from $|\mathcal{P}|$ to find the expected number of undetected superiors:

$$|\mathcal{P}| - \frac{M(|\mathcal{P}|+1)}{M+1} = \frac{|\mathcal{P}|(M+1) - M(|\mathcal{P}|+1)}{M+1} \tag{6.7.6}$$

$$= \frac{|\mathcal{P}| - M}{M+1} \tag{6.7.7}$$

We fill in $|\mathcal{P}| = 14301$ and $M = 1000$, yielding a deficit of about 13.29 compounds that are superior but not detected by random partial screening.



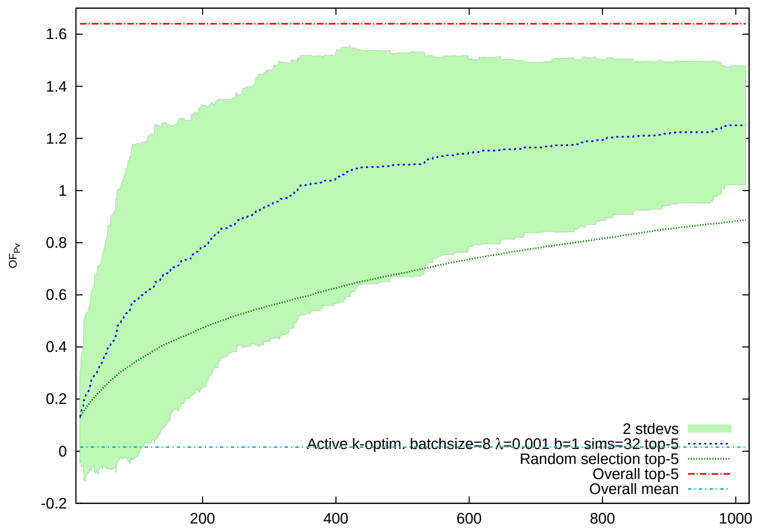
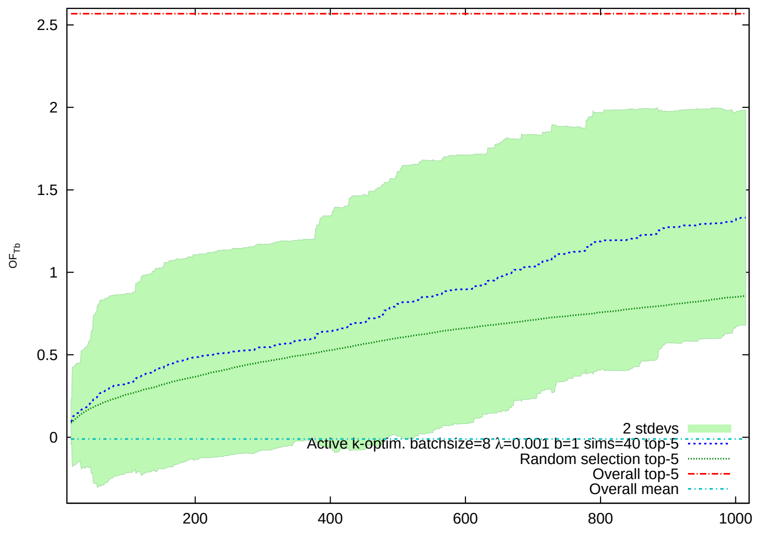


Figure 6.3: The value of $\|OF_{Pf}\|_{\text{best}-5}$ (top) and $\|OF_{Pv}\|_{\text{best}-5}$ (bottom) discovered after screening a given number of compounds, using optimistic active k-optimization with an optimism level of 1. Simulation for experiment batches of 8 compounds.



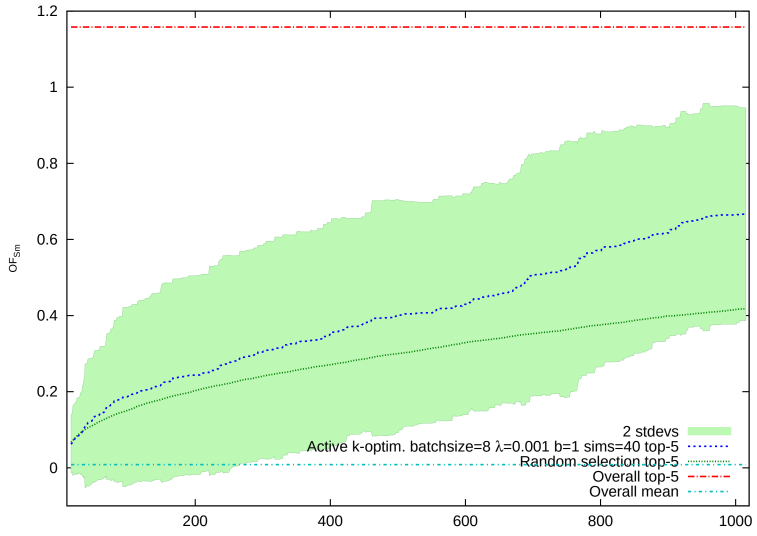
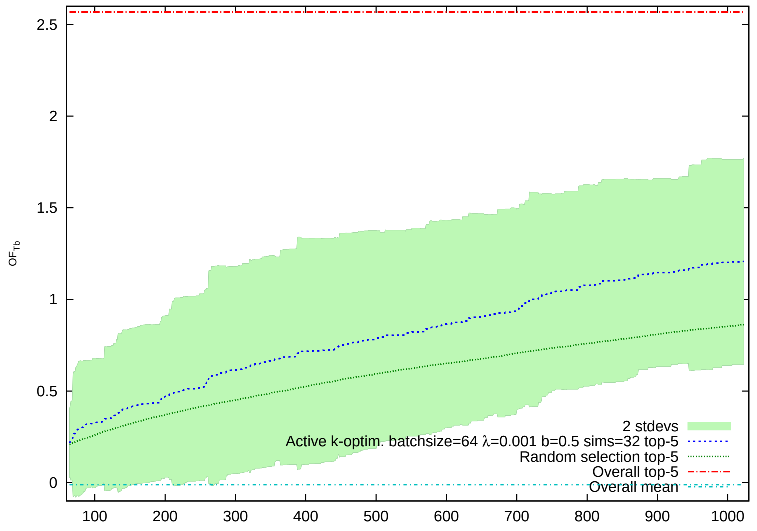


Figure 6.4: The value of $\|OF_{Tb}\|_{\text{best}-5}$ (top) and $\|OF_{Sm}\|_{\text{best}-5}$ (bottom) discovered after screening a given number of compounds, using optimistic active k-optimization with an optimism level of 1. Simulation for experiment batches of 8 compounds.



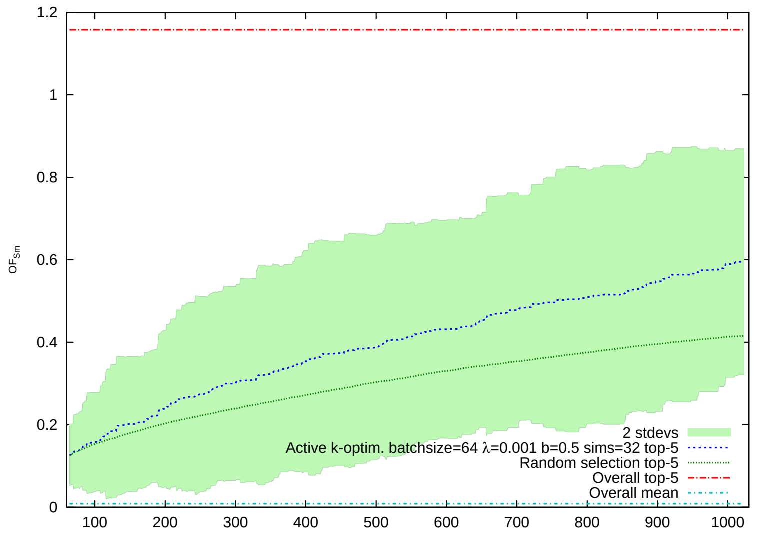


Figure 6.5: The value of $\|OF_{Tb}\|_{\text{best}-5}$ (top) and $\|OF_{Sm}\|_{\text{best}-5}$ (bottom) discovered after screening a given number of compounds, using optimistic active k-optimization with an optimism level of 0.5. Simulation for experiment batches of 64 compounds.

6.7.4 Results

Figures 6.3 and 6.4 show the behaviour of the optimistic optimization procedure for small batch sizes of 8 compounds. The random bootstrap for these simulations was 16 compounds. Each graph shows a yeast strain containing the DHFR from a different pathogen. The finely dotted, steadily rising line shows the result that can be expected from mass screening in a random order. The coarser dotted line above shows the performance of the compounds discovered by the optimization procedure. The shaded area indicates two standard deviations above and below this result[7], as it varies for different bootstraps. Any given large collection of compounds is likely to have on average as much effect on Human DHFR as on DHFR from a pathogen, thus the mean $\|OF\|_{\text{best}-5}$ over all compounds is near zero for all assays. Evidently, some pathogens are easier targets than others: there is considerable variation in the maximum $\|OF\|_{\text{best}-5}$ in HitFinder, as well as in the ease with which the optimization algorithm 'escapes' from the mass screening performance.

Figures 6.5 shows the behaviour of the same procedure for large batch sizes, consisting of 64 compounds each. Since a large batch will automatically contain a larger amount of diversity per experiment, we choose a smaller level of optimism ($b = 0.5$). Note that the vertical scale differs between Figures 6.4 and 6.5. For the investigated budgets of up to 1024 (64 random bootstrap compounds + 15 experiments of 64 compounds), there is only a remarkably low degradation of performance versus the setting in where one can pick a new batch every 8 compounds, depicted in Figure 6.4.

Table 6.1 shows the number of missed superior compounds after screening 1000 compounds (including the bootstrap) out of 14301. Random partial screening leaves undetected many more top-valued compounds, than does active k-optimization. The magnitude of the difference depends of course on the assay. Again there seems to be some tentative degradation of performance when increasing the batch size from 8 to 64.

On average, when screening 1000 compounds with batches of size 8, the optimization algorithm finds the fifth best compound in the library. From Equation 6.7.7 we know that random screening requires the testing of 2859 compounds to obtain the same quality, a nearly threefold greater expenditure.

[6]First choose k, then choose the remaining $M - 1$ numbers from the list $1, 2, \ldots, (k - 1)$.
[7]The shaded area is four standard deviations high.

Pathogen	Pf	Pv	Tb	Sm	Tb	Sm
Batch size	8	8	8	8	64	64
Optimism level k	1	1	1	1	0.5	0.5
Optimistic optimization	6.4	3.3	3.1	3.2	4.6	4.6
Random screening	13.3					

Table 6.1: Number of missed superior compounds after screening 1000 compounds.

6.8 Future library expansion

The library will be extended with the Johns Hopkins Clinical Compound Library (JHCCL) (Chong et al. 2006), a small library exclusively containing drug compounds approved by the US Food and Drug Administration (FDA), by a similar institute in another country, or undergoing phase two clinical trials. The number of single-compound substances (of which the structural formula is available) is 1,700 — significantly lower than the number in (Chong et al. 2006). Screening data from the HitFinder library was modeled by Support Vector Regression (using SVM-Light) and NSPDK (Chapter 3) over augmented graphs (Chapter 2). The comparison between the a priori predictions for JHCCL and the wet lab measurements will provide an independent in vitro validation of the NSPDK QSAR method.

6.9 Conclusions

The Robot Scientist Eve provides an excellent platform for the development of new QSAR modeling techniques, as well as machine learning, planning, and optimization algorithms.

NSPDK was used for the first time in a wet lab to select 11 compounds for purchase and screening against *Plasmodium vivax*. A new strong hit was discovered, even though active compounds are exceedingly rare and the selection was very dissimilar to the hits in the training set. A further three compounds showed weak activity.

We have implemented an active k-optimization algorithm for the selection of compound batches during cherry picking screening. The algorithm seems to work reasonably well: it obtains good compounds after fewer experiments than unguided screening, according to simulations with all data that Eve has obtained so far. Vice versa, there are only very few compounds left in the library that are superior to the best compound detected in 1000 experiments.

Using smaller batches, i.e. executing more cycles of the discovery loop, may allow discovery of top compounds in fewer experiments, but in any case the gain is gradual rather than dramatic for an eightfold batch size reduction.

When selecting small batches of experiments, the algorithm requires nearly three times fewer experiments than random screening to find the fifth best compound in the library.

Chapter 7

Optimization of cellulose acetate nanofiltration membranes

When the well's dry, we know
the worth of water.

Benjamin Franklin

Nanofiltration and reverse osmosis membranes can remove trace contaminants of low molecular weight from water that cannot be removed efficiently by conventional purification treatments. The performance of a membrane depends on several parameters involved in its synthesis. These include compositional (polymer concentration, solvent) as well as non-compositional parameters (temperature, annealing time). We used a genetic algorithm to direct membrane synthesis towards better separation of the targeted compounds, combined with a useful flux.

High throughput filtration experiments were carried out to evaluate the capability to retain ibuprofen in water. Ibuprofen is one of the smallest relevant micropollutants in drinking water. Membranes with up to 96% ibuprofen retention and adequate permeability were obtained, which also showed competitive NaCl retention and twice the permeability compared to membranes prepared via classical line search optimization.

We also explore an alternative optimization method, based on nonlinear Gaussian process modeling and the expected improvement criterion, the basics of which were covered in Chapter 5.

This chapter is based on (Cano Odena et al. 2011) and (Cano Odena et al. 2008).

7.1 Introduction

The ever growing worldwide water demand, together with stricter regulations for potable and waste water, leads to the need for better cleaning technologies to reduce the concentration of micropollutants (pharmaceuticals, endocrine disrupting compounds, etc) in water streams. These micropollutants are mostly of anthropogenic origin. Their presence in water has an impact on environmental and human health. The prioritisation of organic micropollutants removal from surface and ground water is motivated by different criteria, such as their toxicity and their presence in drinking water (Verliefde 2008).

Pressure-driven membrane technologies are well suited to remove trace contaminants. The term nanofiltration (NF) is used for the filtration of particles of around 1 nanometer, whereas the term reverse osmosis (RO) is used at pore sizes of around 0.1 nanometer. Cellulose acetate (CA) is a common polymer used for NF and RO membranes (Morão et al. 2005, Hofman et al. 1997). It has been commercially available since the 70s. (Sourirajan 1977) CA is inexpensive, presents relatively good resistance against chlorinated agents commonly used to disinfect water and is obtained from sustainable sources. However, some challenges still go unsolved, such as the need of an improved chemical stability and a high rejection of organic compounds combined with high water fluxes. Asymmetric membranes, consisting on a thin dense layer that determines the selectivity above a porous sublayer acting as support and providing mechanical stability, are interesting for these applications. (Haddada et al. 2004) They are commonly prepared via phase inversion, which comprises the controlled transformation of a thermodynamically stable polymeric solution into a solid porous phase (Mulder 1996). The final performance depends on multiple factors, including the composition of the polymeric solution (solvents, polymer concentration, additives) and non-compositional parameters at the level of the membrane synthesis process and post-treatment (evaporation time, temperature, annealing time) (Vandezande et al. 2008, Idris et al. 2002).

The optimization of the multiple parameters is complex as well as time and effort consuming. Genetic Algorithms (GA) are stochastic search techniques inspired by the principles of evolution and natural selection found in nature. The successive generations of experiments are created by applying evolutionary operators (mutation and cross-over). A membrane that is experimentally found to be more successful will have more offspring and more variants in the following generation of experiments. Populations thus evolve in a self-adaptive way towards the optimal solution (Goldberg 1989). Genetic algorithms have already been used in the pharmaceutical industry (Casault et al. 2007), material development (Hoogenboom et al. 2003), and catalysis (Casault et al. 2007) leading to successful implementation. These tools have also proven to be extremely useful in membrane technology to develop better performing membranes, directing membrane composition towards improved separation properties (Gevers 2005, Vandezande et al. 2009, Bulut et al. 2006). In such an approach, it is possible to obtain maximum output while reducing time and material consumption (Vandezande et al. 2009). Also they have been used to select the operating variables of the process to optimize the performance of the membrane system (Murthy and Vengal 2006).

Despite their potential, the use of these optimization strategies would be extremely time and material consuming. The availability of high throughput (HT) experimentation enables rapid and accurate collection of large datasets, essential for the implementation of combinatorial synthesis, together with miniaturization (cost and waste reduction) (Vandezande et al. 2005, Zhou et al. 2009).

In this chapter, we consider the optimization of CA-based NF/RO membranes to be applied for salt and micropollutants removal in aqueous streams. The influence of both compositional and, for first time, non-compositional parameters will be explored by using a GA. We will benchmark the membranes for ibuprofen retention from water. Ibuprofen is a non-steroidal anti-inflammatory drug. It is one of the smallest molecules of relevant micropollutants currently present in drinking water (Verliefde et al. 2007). Its successful removal may also indicate removal of all other micropollutants present in the water. Moreover, ibuprofen is the third most consumed pharmaceutical worldwide (Buser et al. 1999). Although its concentration in drinking water is normally below the ‘Human Health Limit’, a general concern exists due to the lack of detailed knowledge about the potential mixture toxicity, which occurs for combinations of certain pharmaceutical compounds that lead to health risks despite being present in very low concentrations (Verliefde et al. 2007). Finally, for selected cases, the membrane performance will be also evaluated for NaCl retention in water in order to compare it with the performance obtained via a classical optimization strategy.

7.2 Methods

7.2.1 Membrane synthesis search space

The membrane solutions consist of CA dissolved in a mixture of solvents and non-solvents in a variable ratio. The components were selected based on the literature (Duarte et al. 2006, Lonsdale 1972). The polymer content ranges from 12 to 25 wt% (percent of mass) and the methanol content between 0 and 25 wt%. The concentration of acetone was kept constant at 20 wt%. Dioxane completes the composition up to 100 wt%.

The membranes are prepared by depositing a film of the polymeric solution on top of a support. After a certain evaporation time (30, 60, 90, or 120 sec), the films are immersed in a coagulation bath. Finally, a thermal annealing treatment follows, by immersing the membranes during 2, 6, 10, or 14 minutes in a water bath at constant temperature (65, 70, 75, 80 or 85 °C). All three non-compositional parameters were taken to be discrete variables because a continuous range of values was deemed impractical for experimentation.

A membrane design is thus specified by an array of five values: two compositional parameters (CA and methanol concentration) and three non-compositional parameters (evaporation time, annealing time, and annealing temperature). The design space is only five-dimensional as the concentration of dioxane is a dependent parameter due to the constraint to have a composition total of 100 wt%.

7.2.2 High throughput filtration experiments

Membrane performance was evaluated in dead-end filtration experiments of a feed solution of 5 mg/l ibuprofen in water at 40 bars. They were carried out by using a custom designed High Throughput module (Figure 7.1). It allows the simultaneous testing of 16 membranes. All experiments were carried out in duplicate. If the relative standard deviation was higher than 10%, a third replicate was tested. Permeabilities ($L/m^2h\ bar$) were measured directly. Retentions were calculated as (1-Cp/Cf)*100% where Cf and Cp refer to the solute concentration of the initial feed and of the permeate respectively.

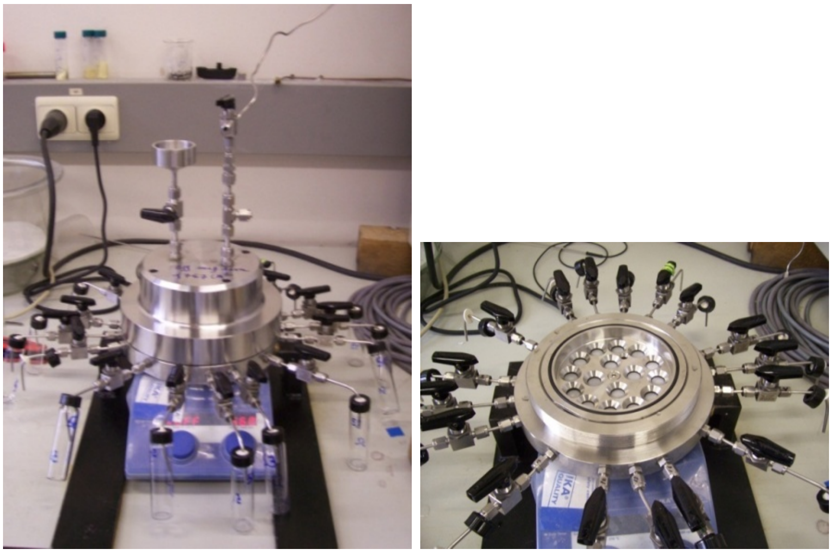


Figure 7.1: High throughput apparatus module for dead-end filtration experiments: overview (left); detail of the positions of the membranes in the module (right).

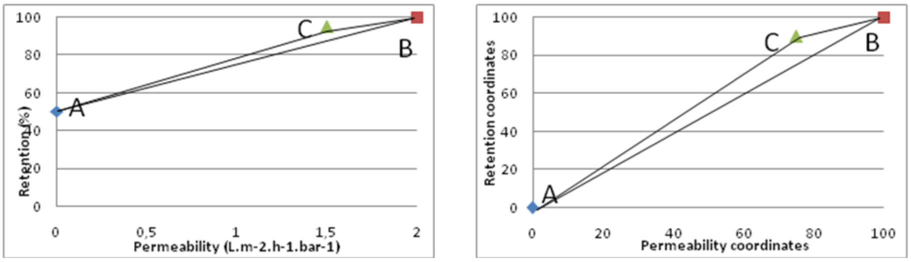


Figure 7.2: Permeability-retention coordinate conversion: (a) Absolute position of the threshold (A), target (B) and an example data point (C); (b) representation in the two-dimensional coordinate space with coordinates in the range [0,100].

7.2.3 Evaluation of membrane performance

Membrane performance was evaluated with an objective function (OF) that combines the permeability (P) and the retention (R). The threshold retention

($R_{threshold}$) was 50 % (A). The target performance (R_{target}) was defined as 100 % solute retention and a water permeability (P_{target}) of 2 $L/m^2h\ bar$ (B) (Figure 7.2a). In order to adjust the weight of each component in the OF, the measured values of permeability ($P_{measured}$) and retention ($R_{measured}$) of the membranes were transformed to new coordinates ranging from 0 to 100 (C_1, C_2) according to the following equations:

$$C_1 = \min\left(100, \left(\frac{P_{measured}}{P_{target}}\right) \times 100\right) \tag{7.2.1}$$

$$C_2 = \max\left(0, \left(\frac{R_{measured} - R_{threshold}}{R_{target} - R_{threshold}}\right) \times 100\right) \tag{7.2.2}$$

The OF values were calculated by the subtraction of the distance BC from AB in the coordinate space, following the formula:

$$OF = AB - BC = \sqrt{((B1 - A1)^2 + (B2 - A2)^2} - \sqrt{(B1 - C1)^2 + (B2 - C2)^2} \tag{7.2.3}$$

Where the coordinates of A, B, and C after transformation by Equations 7.2.1 and 7.2.2 are (0, 0), (100, 100), and (C_1, C_2) respectively. The calculation of OF in this particular case is:

$$OF = AB - BC = \sqrt{100^2 + 100^2} - \sqrt{(100 - C1)^2 + (100 - C2)^2}$$

The closer the measured and target values are, the higher will the OF value be (Figure 7.2b). Figure 7.4 shows isometrics for the OF.

7.2.4 Genetic algorithm

The combinatorial optimization of the membranes was carried out by applying a GA, saving the time and effort required for a complete systematic one-by-one screening of each parameter. The algorithm was coded in a Microsoft Excel spreadsheet using Visual Basic for Applications. An overview of the GA steps is presented in Figure 7.3.

Population size The population size was selected to be a multiple of 16 because the HT setups permit 8 and 16 simultaneous membrane synthesis and testing simultaneously. In related work, a population size of 64 was selected for 8 parameters (Bulut et al. 2006). Since the total number of parameters to be optimized here is only 5, a population size of 48 should be adequate.

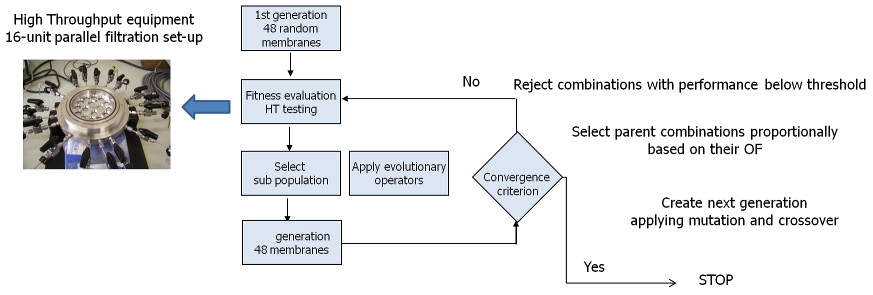

High Throughput equipment
16-unit parallel filtration set-up
1st generation 48 random membranes
Fitness evaluation HT testing
Select sub population
Apply evolutionary operators
generation 48 membranes
Convergence criterion
No
Reject combinations with performance below threshold
Select parent combinations proportionally based on their OF
Create next generation applying mutation and crossover
Yes
STOP

Figure 7.3: Overview of the genetic algorithm optimization steps.

Initialization The membrane designs of the first generation were generated randomly. That is, for each parameter, a value was randomly selected in the ranges presented in Section 7.2.1. Note that although the independent parameters were each drawn from a uniform distribution, the resulting distribution of the dependent parameter dioxane is not uniform.

Evolutionary operators The parents for every next generation were selected with the roulette wheel method, proportional to their fitness (OF). In the present study, only crossover and quantitative mutation were applied. Qualitative mutation was not considered as it involves the addition or elimination of one of the components. In our setting, the exclusion of one of the parameters would lead to an unfeasible combination. Crossover creates new individuals by exchanging a fragment of the digits between two individuals at a random position in the sequence of parameters. In quantitative mutation (hereafter referred to as mutation) the value for one randomly selected parameter changes. If the parameter is continuous, Equation 7.2.4 is applied:

$$x_i^{new} = (1+t) \cdot x_i^{old} \tag{7.2.4}$$

where t is a random number that controls both the direction and relative size of the mutation, sampled uniformly from the range [-0.5,+0.5].

The mutation method was inspired by (Wolf et al. 2000). However, in the present study, in order to improve coverage of the design space, values cannot just mutate by ±50%, but also by any smaller fraction. The compositional parameters were rounded to the nearest integer weight percentage, to avoid unfeasibly small variations of the compositional ratios, potentially exceeding the precision of the experimental method and apparatus. Whenever x_i^{new} is out of range, a new t is drawn and the procedure is repeated.

The mutation of a non-compositional, discrete parameter is calculated with Equation 7.2.5. The value changes into the nearest higher or lower value, depending on the direction, determined by a random variable s which can be 1 or -1.

$$x_i^{new} = x_i^{old} + \; s\triangle x_i^{old} \tag{7.2.5}$$

When the result of the mutation exceeds the range of the parameter, the opposite extreme value is assigned.

The frequency to apply each operator (W_i) depends on the relationship between the values of OF_{best} and OF_{mean}, as follows:

$$W_{crossover} = \frac{B \times \; \mathrm{OF}_{mean}}{\mathrm{OF}_{best}} \tag{7.2.6}$$

$$W_{mutation} = 1 - W_{crossover} \tag{7.2.7}$$

B is a control parameter and is set to 1.

7.3 Results and discussion

7.3.1 Observations for the four generations

First generation Some of the membranes in the first generation were defective, with extremely high permeability and practically no retention.

The membranes with the highest permeabilities generally had a relatively low polymer contents (12-15 wt%), which is in agreement with the literature (Liu and Bai 2006, Saljoughi et al. 2009). However, membranes with a low polymer concentration do not always have a high permeability.

Second generation All 48 second-generation membranes could be synthesized, but two of them presented defects and were assigned an OF value of zero.

The OF_{best} value in the second generation was 63.4, lower than in the first one. The same effect has been observed in previous work (Vandezande et al. 2009). The OF_{mean} was 31.8, slightly lower than the first generation value (34.9). Clearly, the overall performance of the population has not yet improved.

Third generation All of the 48 membranes could be cast but 6 showed defects and lack of stability. The OF_{best} and OF_{mean} were 89.5 and 37.7 respectively, both higher than the previous generations.

Fourth generation The probability of applying the crossover operator, which creates more diversity in the population, was lower than in the previous generations. All 48 membranes could be tested and none presented defects.

According to a Student t-test, the OF values in the fourth generation are higher than random sampling (first generation) at a 5% significance level. On average, the membranes in this generation featured higher retention but lower permeability than the third generation.

Table 7.1: Frequency of the operators, membrane performance and OF values of the four generations.

Generation	$W_{crossover}$	$W_{mutation}$	OF_{mean}	OF_{best}	Best membrane	Retention (%)	Permeability	$\frac{OF_{mean}}{OF_{best}}$	$OF_{best} - OF_{mean}$
1	–	–	34.9	86.2	M1-8	87.73	1.01	0.405	51.3
2	0.594	0.406	31.8	63.4	M2-12	90.35	0.49	0.502	31.6
3	0.496	0.504	37.7	89.5	M3-47	82.50	1.23	0.366	56.6
4	0.635	0.365	43.2	85.3	M4-37	77.85	1.31	0.506	42.1

7.3.2 Overall results

The summary of the results of the 4 generations is presented in Table 7.1. Detailed results are presented in Figure 7.4. The first two generations can be characterized as tending towards high permeability, whereas the later generations generally achieved high retention.

The distribution of the membrane performance over the 4 generations is presented in Figure 7.5. It shows clearly that there is a general increase in the OF values in the 4^{th} generation. The highest OF value corresponds to a membrane of the 3^{rd} generation. Previous work showed a faster progress (Vandezande et al. 2009). A possible explanation is that we did not discard membranes with sub-treshold retention, but rather assigned a low OF.

The 10 membranes with the highest OF values of all 192 membranes are presented in Table 7.2. In general, these membranes present ibuprofen retentions below 90 %. However, the 7^{th} membrane in the ranking (M2-12) has a very good retention (90 %) combined with a reasonable permeability (0.49 $L/m^2h\ bar$),

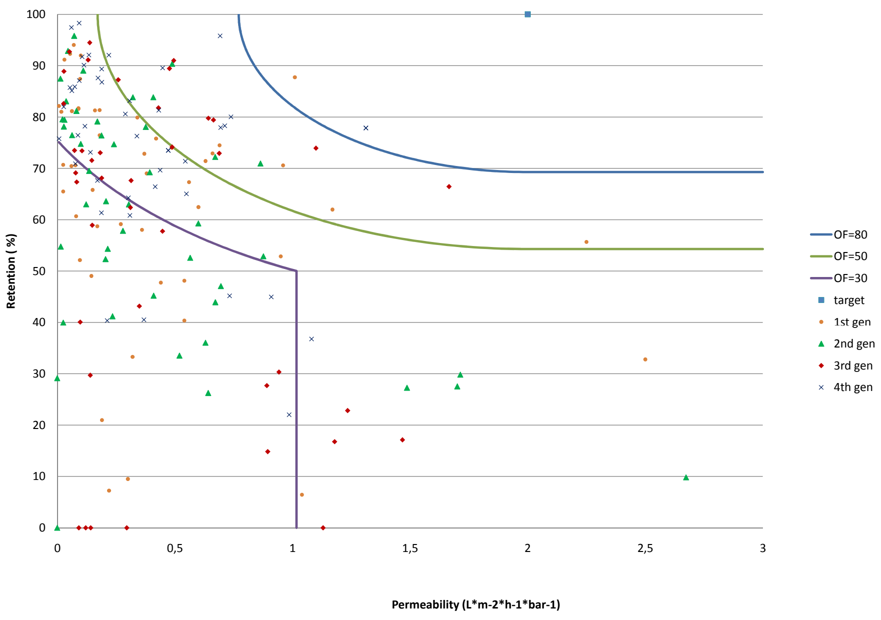


Figure 7.4: Water permeability and ibuprofen retention of all membranes of the four generations. The target point and three objective function isometrics are also shown.

being comparable to the values for CA-based membranes for RO applications (Duarte et al. 2006).

In hindsight, the permeability target in this optimization was too ambitious; therefore high permeabilities were obtained at the expense of retention.

7.3.3 Application to water desalination

No prior literature on CA membranes for ibuprofen separation exists, so there is no direct benchmark. In order to compare the performance of the obtained membranes with CA membranes in the literature, NaCl filtration experiments were carried out.

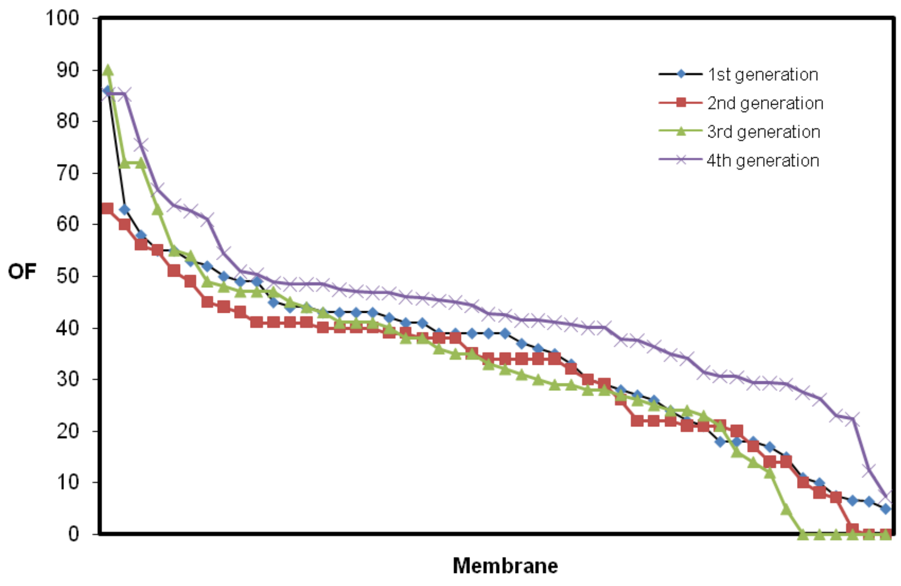


Figure 7.5: Change in the distribution of membrane quality over the four generations.

A total of 5 membranes (the best membrane in terms of OF, and the two best ones for retention and permeability) were selected to perform 5g/l NaCl filtration experiments in order to compare with earlier CA-membrane optimizations reported in the literature. The results are presented in Table 7.3.

Membrane M3-47 retains 83% NaCl, which is as high as the best CA membrane reported in the literature (Duarte et al. 2006), while its water permeability is twice as high (1.2 relative to 0.6 $L/m^2h\ bar$). Duarte et al. prepared 45 different membranes and performed a classical optimization by fixing all but one parameter and screened parameters one by one. None of those membranes reached performances (permeability combined with selectivity) as good as some of the best membranes in our project. This indicates that there are parameter combinations that lead to superior performance, which were missed by the traditional line search optimization approach.

Table 7.2: Selection of the 10 membranes with the highest OF values. Results are the average of 3 replicates at 40 bar, room temperature.

Membrane	CA (wt%)	Methanol (wt%)	Dioxane (wt%)	Acetone (wt%)	Evaporation time (s)	Annealing time (min)	T ($°C$)	OF	Permeability ($L/m^2h\ bar$)	Retention (%)
M3-47	19	7.5	54	20	60	2	75	89.5	1.2	82.5
M1-8	13	14	52	20	30	14	85	86.1	1.0	87.7
M4-20	19	7.5	54	20	30	2	75	85.3	1.3	77.8
M3-21	18	12	50	20	60	6	80	72.3	0.9	79.4
M3-39	19	7.5	54	20	120	2	75	71.7	1.1	73.4
M4-11	12	23	45	20	30	10	75	66.7	0.7	80.00
M2-12	20	13	47	20	60	14	80	63.4	0.49	90.35
M3-11	13	12	55	20	30	14	80	62.9	0.49	89.45
M1-4	13	6.2	60.8	20	60	6	75	62.9	0.96	70.57
M4-46	12	17	51	20	60	14	70	62.7	0.69	77.95

7.4 Active optimization

In this section, we describe an alternative optimization method, for which we currently lack sufficient experimental results[1].

7.4.1 Active optimization of membrane designs

As an alternative to the genetic algorithm, we have developed a version of the optimization algorithm described in Chapter 5 to select membranes for testing. As in Chapter 6, a key difference with the original algorithm is the requirement to select batches of experiments rather than one at a time. Unlike the experiments performed by the robot scientist Eve, the synthesis of membranes is performed by humans. It is more expensive and more time consuming to synthesize a

[1]Unfortunately, due to several factors outside our control, only three batches of eight membranes have been synthesized, including the bootstrap batch.

Table 7.3: Desalination performance of key membranes. Results are the average of 3 replicates at 40 bar, room temperature.

Membrane	NaCl retention (%)	Permeability ($L/m^2h\ bar$)	OF	Comments
M3-47	82.55	1.2	89.53	highest OF
M4-8	79.69	0.06	44.29	2nd highest retention
M4-28	41.70	0.09	45.99	highest retention
M4-47	37.82	37.7	41.42	highest permeability
M2-4	26.08	11.4	41.42	2nd highest permeability

membrane than for Eve to test a single compound (amortized over one or more microplates). This allowed us to spend more computation time to choose optimal experiments.

We opted for the maximum expected improvement criterion (MEI), which readily generalizes to batch selections. Recall its definition (Equation 5.7.3):

$$\mathbb{E}(gain) = \mathbb{E}\left[\max\left(0, \left(t_{N+1} - t_{\#(k,N)}\right), \ldots, \left(t_{N+n} - t_{\#(k,N)}\right)\right)\right]$$

However, the generalization is not trivial to compute.

A first step in our algorithm is to rescale all membrane parameters to the same numerical range.

The physical properties of a CA membrane are known to depend non-linearly on the synthesis parameters. To allow our model to capture the non-linear behaviour, we use the popular exponential kernel for representing the similarity of membrane designs:

$$k(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2}\right) + \sigma\delta(x,y) \tag{7.4.1}$$

where $\sigma\delta(x,y)$ is the regularization term.

7.4.2 Computation of the expected improvement of a batch of experiments

We assume that there is a linear model in the vector space induced by the kernel:

$$OF(x_i) = w^\top \phi(x_i) \tag{7.4.2}$$

We know the posterior distribution of w under a Gaussian process model (Equation 5.3.16) and we will here write its mean as $\bar{w} \in \mathbb{R}^d$ and its covariance matrix as $\Sigma \in \mathbb{R}^{d \times d}$. We can now rewrite Equation 5.7.3 as:

$$\mathbb{E}(gain) = \int_w \max_{i=0 \dots n} (w^\top \phi(x_i) - b_i) \frac{\exp\left(-\frac{1}{2}(w - \bar{w})^\top \Sigma^{-1} (w - \bar{w})\right)}{(2\pi)^{d/2}\sqrt{|\Sigma|}} dw \tag{7.4.3}$$

where:

- vectors $\phi(x_i) \in \mathbb{R}^d$, $i = 1 \dots n$
- $b_i = t_{\#(k,N)} \in \mathbb{R}$, $i = 1 \dots n$.
- $\phi(x_0) = \overrightarrow{0}$, $b_0 = 0$

Since we want to find $\arg\max_{x_1,\dots,x_n}(\mathbb{E}(gain))$, we can drop the constant factors:

$$\mathbb{E}(gain) \propto \int_w \max_{i=0 \dots n} (w^\top \phi(x_i) - b_i) \exp\left(-\frac{1}{2}(w - \bar{w})^\top \Sigma^{-1} (w - \bar{w})\right) dw \tag{7.4.4}$$

We need to circumvent the problem that d can be very high — for the exponential kernel it even degenerates to infinity! To this end, we apply a coordinate transformation[2] $\phi(x_i) \to x'_i$ such that all coordinates other than the first n are 0, for all x'_i. Then, we need only to integrate over the first n dimensions. With Q a suitable $d \times n$ matrix of full rank, we can substitute

$$\phi(x_i) = Qx'_i$$

$$w = Qw'$$

$$\bar{w} = Q\bar{w}'$$

$$\Sigma = Q\Sigma' Q$$

$$b_i = b'_i$$

[2]For brevity we write x'_i rather than $\phi'(x_i)$.

Equation 7.4.4 then becomes

$$\mathbb{E}(gain) \propto \int_{w'} \max_{i=0\dots n} ({w'}^\top x'_i - b'_i) \exp\left(-\frac{1}{2}(w' - \bar{w}')^\top {\Sigma'}^{-1}(w' - \bar{w}')\right) dw' \tag{7.4.5}$$

An appropriate transformation matrix Q is obtained from the QR-decomposition of $\Phi = [\phi(x_1)\phi(x_2)\dots\phi(x_N)]$. The matrix R will then contain the non-zero new coordinates x'_i. We don't need Q itself — fortunately, for its number of rows d is infinite. Since the Gramian $\Phi^\top\Phi$ is symmetric and positive semi-definite, it is sufficient to compute its Choleski decomposition $\Phi^\top\Phi = LL^\top$ with L a lower triangular matrix. We can then use the columns of $L^\top$ as the new representations x'_i.

A potential issue is that the x_i are likely to be not randomly distributed in space, but can probably be grouped in clusters of vectors pointing in similar directions. This may have a negative influence on the stability of the computation of the decomposition.

We can eliminate Σ' by substituting

$$w' = \sqrt{\Sigma'} w_*$$

$$\bar{w}' = \sqrt{\Sigma'} \bar{w}^*$$

$$x'_i = \frac{C x^*_i}{\sqrt{\Sigma'}}$$

$$b'_i = C b^*_i$$

where C is a constant meant to keep $x^*_1 \in [-1, +1]^n$. We then get

$$\mathbb{E}(gain) \propto \int_{w_*} \max_{i=0\dots n} (w_*^\top x^*_i - b^*_i) \exp(-\frac{1}{2}(w_* - \bar{w}^*)^\top (w_* - \bar{w}^*)) dw_* \tag{7.4.6}$$

We can further substitute

$$w_* = w_\# + \bar{w}^*$$

$$b^*_i = b^\#_i + \bar{w}^* x_i$$

to get

$$\mathbb{E}(gain) \propto \int_{w_\#} \max_{i=0\dots s} (w_\#^\top x^*_i - b^\#_i) \exp\left(-\frac{1}{2} w_\#^\top w_\#\right) dw_\# \tag{7.4.7}$$

Symbolic integration of Equation 7.4.7 is very difficult. A viable numeric integration option was proposed by Bart Vandewoestyne. He also provided an elegant Monte Carlo integration implementation in Matlab (Vandewoestyne 2008). We optimized the code by specializing it for our purposes. The result is described in Algorithm 3.

Algorithm 2 sketches the optimization algorithm. The accuracy achieved by Monte-Carlo integration is proportional to the logarithm of the number of samples. We also developed a parallelized version for the supercomputer of our university using the Matlab Distributed Computing Engine. The evaluation of candidate membranes is trivially distributed over multiples nodes with negligible synchronization traffic. (Some computation could be traded off against synchronization by finer-grained cycles of attrition and approximation refinement.)

To bootstrap the optimization cycle, a first batch is composed with a Latin hypercube design (McKay et al. 1979). This is a scheme for stratified sampling from a hyperrectangle in a multidimensional space. The idea is to split each dimension of the hyperrectangle in as many identically sized intervals as there are samples to be taken. Each interval receives exactly one sample. Algorithm 4 gives a straightforward adaptation of Latin hypercubes for discrete design spaces.

7.5 Conclusions

An evolutionary search strategy together with the use of HT experimentation permitted the search of a multi-parameter space in the real-world application domain of filtration membrane synthesis. For the first time also non-compositional parameters of the synthesis process have been included in a GA-based optimization procedure.

Over the four generations, an improvement of the overall performance was observed. High-performance membranes were thus developed for the retention of micropollutants from aqueous streams with a very good retention (up to 96%) of the small target compound ibuprofen, combined with a permeability of 0.7 $L/m^2h\ bar$, which is in the normal range of CA-based reverse osmosis membranes.

Moreover, a membrane design was found with the same NaCl retention and twice the permeability value reported in the literature.

We have also generalized the optimization algorithm introduced in Chapter 5 to work with arbitrary nonlinear kernels, if the training examples are not too

Algorithm 2 MEI optimization with batch selection

```
1: **Given**: membrane design space $\mathcal{D}$, the $N$ already tested membranes $X_N$, their permeabilities $P_N$, and their retention $R_N$
2: **Return**: the next 8 membranes to be tested and their expected OF
3: $\mathcal{P} \leftarrow$ enumerate all possible membrane designs in $\mathcal{D}$
4: $OF_N \leftarrow$ Equation 7.2.3
5: Rescale all features such that $\forall i, j : X_{ij} \in [-\sqrt{1/5}, \sqrt{1/5}]$ hence $max(|X_i|) = 1$
6: **for all** $X_i \in \mathcal{P}$ **do**
7:     $G \leftarrow$ Gram matrix of the exponential kernel for $X_N \cup \{X_i\}$
8:     $G_{Chol} \leftarrow$ Choleski transform of $G$
9:     $X_i^{Chol} \leftarrow$ coordinates of $X_i$ in $G_{Chol}$
10: **end for**
11: Compute $b_i^{\#}$ from Equations 7.4.3–7.4.7
12: $X_{Chosen} \leftarrow \phi$
13: Fit a Gaussian process model to $X_N^{Chol}, OF_N$
14: **for all** $j = 1 \ldots 8$ **do**
15:     **for all** $X_i^{Chol} \in \mathcal{P}^{Chol} \backslash (X_N^{Chol} \cup X_{Chosen})$ **do**
16:         Compute gain with low-resolution Monte-Carlo numeric integration (20,000 samples) using Algorithm 3
17:     **end for**
18:     For the 3000 membranes with the highest gain, compute medium-resolution Monte-Carlo numeric integration (200,000 samples)
19:     For the 100 membranes with the highest gain, compute high-resolution Monte-Carlo numeric integration (2,000,000 samples)
20:     $X_{Chosen} \leftarrow X_{Chosen} \cup$ the membrane with the highest gain
21: **end for**
22: **return** The untransformed design parameters of $X_{Chosen}$ and $\mathbb{E}\left(OF(X_{Chosen})\right)$
```

plentiful. Finally, we have also shown how to construct arbitrarily sized batches of experiments according to the maximum expected improvement criterion.

Algorithm 3 Monte Carlo integration of a function proportional to the expected improvement

```
1: **Given**: feature matrix $X$, integration lower limits $b$, number of samples $s \gg 1$
2: **Return**: approximation for Equation 7.4.7 (accuracy proportional to $\log(s)$)

3: $n \leftarrow$ number of features (rows in $X$)
4: $m \leftarrow$ number of membranes (columns in $X$)
5: $w \leftarrow n$ samples of $\mathcal{N}$
6: $r \leftarrow \max(0, \max_{j=1\dots m}(w_j^\top X_j - b_j))$
7: for all $i = 2 \dots s$ do
8:     $w \leftarrow n$ samples of $\mathcal{N}$
9:     $r \leftarrow r + (\max(0, \max_{j=1\dots m}(w_j^\top X_j - b_j)) - r)/i$
10: end for
11: return $r$
```

Algorithm 4 Latin hypercube for a discrete design space

```
1: Given: number of features $|\{f_i\}|$, all possible values $f_{ij}$ for each feature $f_i$, number of desired output designs $o = 8$
2: Return: a Latin hypercube sample from the design space
3: for all $p = 1 \dots |\{f_i\}|$ do {Take a stratified sample of size $o$ from the uniform distribution over $\{f_{pj}\}$.}
4:     Multiset $S_p \leftarrow \left\lfloor \frac{o}{|\{f_{pj}\}|} \right\rfloor$ copies of all $f_{pj}$.
5:     Add to $S_p$ a simple random sample without replacement of size $o - |S_p|$ from the uniform distribution over $\{f_{pj}\}$
6:     $C_p \leftarrow$ random permutation of $S_p$
7: end for
8: return $C = [C_1 \dots C_{|\{f_i\}|}]^\top$
```

Part III

Conclusions

Chapter 8

Conclusions

To prevent the recurrence of misery is,
alas! beyond the power of man.

(Malthus 1798)

8.1 Summary of contributions

In this thesis, I have explored two avenues where machine learning can help drug discovery: predictive models of in vivo or in vitro effects of molecules, and the selection of efficient experiments based on such models.

In the first part, I have presented methods to improve the predictive power of graph kernel based molecule classifiers. A first approach was the annotation of molecular graphs with functional groups (Chapter 2). All earlier attempts to modify molecular graphs were in the direction of simplification and abstraction. By contrast, we extended the graph representation. The graph kernel based machine learning algorithm can then use both high-level functional, and low-level atomic information, so achieving larger AUCs.

Functional groups often imply very specific configurations of only a few atoms. Mimicking this in a chemically-agnostic graph kernel leads to the notion of neighborhood subgraphs. Atomic neighborhoods (or atomic environments) are not new, in fact the commercial software package Pipeline Pilot contains an implementation. Apart from the comparative study of a number of graph kernels, our contribution (together with Fabrizio Costa) in this area is twofold:

we use neighborhood pairs at given distances from each other, rather than single neighborhoods, and we introduce an efficient method to encode all pairs of atomic neighborhoods in a molecule as a sparse vector. The learning bias of an SVM using these vectors is suitable for a wide range of QSAR tasks. For example, the accuracy of Ames toxicity predictions is as high as the actual in vitro test in the wet lab.

The combined efficiency and predictive power of the NSPDK graph kernel, as applied to molecule classification problems, motivated a larger group of people to devise a novel way to learn from general relational data: transform the data into a graph and apply a graph kernel or feature generator (Chapter 4). The graphicalization is governed by a domain definition language which allows for intensional predicates. The graph kernel and learning algorithm can be selected and configured by the user. Preliminary, yet unpublished experiments indicate great promise for this technique.

In Part II the learned models are a means rather than an end. The end goal of drug discovery research is typically not obtaining an accurate predictive model of the activity of each and every imaginable compound, but rather the invention of a few compound lines that exhibit a superior activity-toxicity trade-off. In Chapter 5, we establish this goal as a new setting in machine learning. We apply surrogate-based optimization to the problem and evaluate how well the different available selection strategies work in the domain of drug screening.

The algorithm is extended to batch selection in Chapter 6, where we also establish its utility in practice through its integration into the robot scientist Eve.

Finally, in Chapter 7 a variant of the algorithm is adapted to the discovery of filtration membranes. For this application, it was necessary to plan batches of several experiments, as well as to compute in instance space rather than feature space.

8.2 General discussion and future work

8.2.1 QSAR

Significant progress has been made in ligand-based QSAR since the introduction of fingerprints, with ever new variations on feature trees, kernels, and other similarity measures. However, the phase of diminishing returns has set in. There is only so much information to be extracted from a given amount of pure ligand data.

The grass is not greener at the target-based side of the fence. No foolproof docking method exists. A single docking method may work very well on one target protein but perform poorly on another — even worse than random selection (McInnes 2007, Figure 1). While the holy grail of drug discovery is to skip the approximative extrapolation over ligands entirely and to simulate the biological system from first principles, this is still a very distant goal. Simulating a full cell system with the complex dynamics of interacting macromolecules, small molecules, membranes and transport effects will remain infeasible for some time. Even the models for the affinity in just a single ligand-target complex (docking) are inadequate. Schneider (Schneider 2010) goes so far as to use the terms "stagnation" and "grossly inaccurate". It is no surprise then, that target-based methods are outperformed by ligand-based methods (McGaughey et al. 2007). Fortunately, Moore's law continues to make computation cheaper, which can be traded in for some additional accuracy or diversity in docking algorithms. Analogous to the use of ensembles in machine learning, it has been recognized in the QSAR literature that it makes sense to use a variation of methods (Sheridan and Kearsley 2002).

At the ligand-based side, further progress is likely to be significant only if more knowledge can be brought in to a problem, other than raw ligand structures. We have taken a step in this direction with the augmentation of molecular graphs using background knowledge on functional groups. There is clearly room for more in this direction. A straight-forward next step would be to take into account that some functional groups are bioisosteric. This can be accounted for with a functional group kernel, or a hierachy of augmentation nodes. A good indication that this may work, is that molecule kernels tend to perform better when using partial charges, which can be seen as an alternative approximation of the same underlying chemical phenomenon. We also plan to upgrade our current kernel to also take 3D shape into account. It is essential for such a method to be either conformation-independent, or at least to support multi-instance learning to allow for multiple conformations.

For fundamental progress, we need to reformulate the problem. Today's drug discovery databases contain not only screening information on one target, but on many targets. These targets are sometimes closely related. The knowledge about these targets is also steadily increasing. Clearly, methods able to take advantage of this information have an edge, be it kernel fusion, transfer learning, statistical relational learning, or some combination thereof. A similar direction has been taken in docking, in the form of learned scoring functions. It would be interesting to pursue such an approach to the QSAR problem.

8.2.2 Optimization

Machine learners entering the QSAR field may find that their work is ignored by drug discovery researchers. The reason for this is that the key problems faced in drug discovery are essentially not a traditional machine learning setting. As we have hopefully made clear in the second part of this thesis, the goal is not as much to create an accurate model, as it is to optimize a function in chemical space. Luckily for the machine learner, this task is daunting without an excellent, well-regularized surrogate model of the function. It is also essential to estimate the reliability of the model in the different regions of chemical space.

A related issue that machine learners need to be aware of, is that the evaluation methodology in drug discovery differs from the evaluation method most commonly practiced in machine learning, internal cross-validation. The drug discovery setting more often than not violates the dogma of machine learning, that training and test sets are identically distributed. In such circumstances, a method is adopted only if it has been proven on independent test sets — from a different region of the chemical space. Such extrapolation is intrinsically hard.

Scaffold hopping is an important capability of virtual screening methods. It is the ability to identify active compounds that have an entirely different scaffold (or basic backbone structure) from the already known actives. Scaffold hopping is the chemist's way not to get stuck in local optima. Sensible global optimization algorithms also have to be designed not to become trapped into local optima. They must explore, not just exploit. Because this behaviour is a key design principle, we expect that global optimization methods as presented in this thesis will do well in terms of scaffold hopping. Unfortunately, no precise, mathematical definition of scaffold hopping exists.

Active k-optimization algorithms have the potential to become a principled alternative to diversity-based screening selections. However, to become widely adopted, a number of improvements need to be made which require further investigations. It should be studied how different approximate GP models (subset of regressors, of data, projected process, ...) influence the optimization loop. This may enable the use of an arbitrary kernel even when confronted with a large amount of training data, rather than having to revert to a low- or moderate-dimensional feature space. Once we have more flexibility in choosing a kernel, the question that immediately emerges is how much effort one should devote to choosing the kernel and learning its hyperparameters?

Next to more powerful models, it is also possible that one could make further gains by devising a strategy that takes into account a budget that is fixed from the start. Exploration can then be more concentrated in the early phase.

From the Eve project, it is clear that efficient selection of large batches of experiments is essential in drug discovery. It is desirable to study alternative heuristics and their impact on the performance of the optimization. In this regard, it is also relevant to investigate the relation to different surrogate-enhanced evolutionary approaches which can typically generate large batches.

It is unclear how well our current approach fits into the “intelligent robot” vision. More explicit knowledge representation and the possiblity to gather types of knowledge, other than pure function approximation would surely fit better. Also from an optimization performance standpoint, breakthrough improvements will require the injection of more background knowledge into the optimization problem. A good first choice is to investigate a possible marriage of transfer learning with surrogate-based optimization. For successful transfer learning, information on the (correlations between) targets will need to be represented.

8.2.3 kLog

kLog is a machine learning language, therefore the horizon of what can be done is very wide. Several theses can be devoted to algorithms for solving the advanced problems that kLog can specify, such as collective classification, collective regression, and learning multiple relations (kLog’s mildly restricted form of structured output learning). Improving the practical usability also has value, for example strategies can be introduced for the automatic selection and tuning of the graph kernel — possibly based on kernel target alignment on small samples. Closer to the core subject of this thesis, the representational capabilities of kLog may be useful to tackle multi-task QSAR learning. It is trivial to represent in kLog highly complex domains of biological studies, with related species, proteins, and ligands. It is probably nontrivial to devise a combination of representation and kernel that reliably outperforms current single-target models. The potential significance for drug discovery of a system with that capability goes without saying.

Bibliography

Ando, H. Y., Dehaspe, L., Luyten, W., Craenenbroeck, E. V., Vandecasteele, H. and Meervelt, L. V.: 2006, Discovering H-Bonding Rules in Crystals with Inductive Logic Programming, *Mol. Pharm.* **3**(6), 665–674.

Bader, R. F. W., Popelier, P. L. A. and Keith, T. A.: 1994, Theoretical definition of a functional group and the molecular orbital paradigm, *Angew. Chem. Int. Edit.* **33**(6), 620–631.

Barker, E. J., Gardiner, E. J., Gillet, V. J., Kitts, P. and Morris, J.: 2003, Further development of reduced graphs for identifying bioactive compounds, *J. Chem. Inf. Comput. Sci.* **43**(2), 346–356.

Ben-David, S., Eiron, N. and Simon, H. U.: 2002, Limitations of learning via embeddings in euclidean half spaces, *J. of Mach. Learning Research* **3**, 441–461.

Benigni, R. and Giuliani, A.: 2003, Putting the predictive toxicology challenge into perspective: Reflections on the results, *Bioinformatics* **19**(10), 1194–1200.

Beume, N., Naujoks, B. and Emmerich, M.: 2007, Sms-emoa: Multiobjective selection based on dominated hypervolume, *European Journal of Operational Research* **181**(3), 1653–1669.

Bishop, C. M.: 2006, *Pattern Recognition and Machine Learning*, Springer.

Blockeel, H., Dehaspe, L., Ramon, J., Struyf, J., Van Assche, A., Vens, C. and Fierens, D.: 2009, The ace data mining system: User's manual. `http://dtai.cs.kuleuven.be/ACE/doc/ACEuser-1.2.16.pdf` (accessed July 2, 2009).

Bohacek, R. S., McMartin, C. and Guida, W. C.: 1996, The art and practice of structure-based drug design: A molecular modeling perspective, *Medicinal Research Reviews* **16**(1), 3–50.

Borgwardt, K. and Kriegel, H.: 2005, Shortest-path kernels on graphs, *in* X. Wu (ed.), *Proceedings of the 5th IEEE International Conference on Data Mining*, Vol. 5, IEEE Computer Society, Houston, Texas, pp. 74–81.

Borgwardt, K. M.: 2007, *Graph kernels*, PhD thesis, Ludwig-Maximilians-Universität München.

Brown, N.: 2009, Chemoinformatics—an introduction for computer scientists, *ACM Comput. Surv.* **41**(2), 1–38.

Bulut, M., Gevers, L. E. M., Paul, J. S., Vankelecom, I. F. J. and Jacobs, P. A.: 2006, Directed development of high-performance membranes via high-throughput and combinatorial strategies, *Journal of Combinatorial Chemistry* **8**(2), 168–173.

Buser, H.-R., Poiger, T. and Müller, M. D.: 1999, Occurrence and environmental behavior of the chiral pharmaceutical drug ibuprofen in surface waters and in wastewater, *Environmental Science & Technology* **33**(15), 2529–2535.

Butina, D.: 1999, Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets, *Journal of Chemical Information and Computer Sciences* **39**(4), 747–750.

Cano Odena, A., Spilliers, M., Dedroog, T., De Grave, K., Ramon, J. and Vankelecom, I.: 2011, Optimization of cellulose acetate nanofiltration membranes for micropollutant removal via genetic algorithms and high throughput experimentation, *Journal of Membrane Science* **366**(1–2), 25–32.

Cano Odena, A., Vandezande, P., Cools, I., Vanderschoot, K., De Grave, K., Ramon, J., De Raedt, L. and Vankelecom, I.: 2008, Comparison of multi-parameter optimization strategies for the development of nanofiltration membranes for salt and micropollutants removal, *Proceedings of the International Congress on Membranes and Membrane Processes*, Honolulu, Hawaii.

Casault, S., Kenward, M. and Slater, G. W.: 2007, Combinatorial design of passive drug delivery platforms, *International Journal of Pharmaceutics* **339**(1–2), 91–102.

Ceroni, A., Costa, F. and Frasconi, P.: 2007, Classification of small molecules by two- and three-dimensional decomposition kernels, *Bioinformatics* **23**(16), 2038–2045.

Chong, C. R., Chen, X., Shi, L., Liu, J. O. and Sullivan, D. J.: 2006, A clinical drug library screen identifies astemizole as an antimalarial agent, *Nature Chemical Biology* **2**(8), 415–416.

Coello Coello, C.: 2006, Evolutionary multi-objective optimization: a historical view of the field, *Computational Intelligence Magazine, IEEE* **1**(1), 28–36.

Cohn, D., Ghahramani, Z. and Jordan, M. I.: 1996, Active Learning with Statistical Models, *Journal of Artificial Intelligence Research* **4**, 129–145.

Costa, F. and De Grave, K.: 2010, Fast neighborhood subgraph pairwise distance kernel, *Proceedings of the 27th International Conference on Machine Learning.*

Courant, R. and Hilbert, D.: 1953, *Methods of Mathematical Physics*, Interscience, New York.

Cox, D. D. and John, S.: 1997, SDO: a statistical method for global optimization, *Multidisciplinary design optimization (Hampton, VA, 1995)*, SIAM, Philadelphia, PA, pp. 315–329.

Cristianini, N. and Shawe-Taylor, J.: 2000, *An Introduction to Support Vector Machines and other Kernel Based Methods*, Cambridge University Press, Cambridge, UK.

De Grave, K. and Costa, F.: 2010a, Augmented molecular graph kernel QSARs, *in* J. Hirst, V. Gillet, J. Goodman, R. Ward, D. Wild and P. Wilett (eds), *Joint Sheffield Conference on Chemoinformatics*, Vol. 5.

De Grave, K. and Costa, F.: 2010b, Molecular graph augmentation with rings and functional groups, *Journal of Chemical Information and Modeling* **50**(9), 1660–1668.

De Grave, K., Ramon, J. and De Raedt, L.: 2008a, Active learning for high throughput screening, *Proceedings of the Eleventh International Conference on Discovery Science*, Vol. 5255 of *Lecture Notes in Computer Science*, pp. 185–196.

De Grave, K., Ramon, J. and De Raedt, L.: 2008b, Active learning for primary drug screening, *in* L. Wehenkel, P. Geurts and R. Marée (eds), *Benelearn 08. The Annual Belgian-Dutch Machine Learning Conference*, pp. 55–56.

De Raedt, L.: 2008a, An introduction to logic, *Logical and Relational Learning*, Springer-Verlag, Berlin Heidelberg, Germany, pp. 17–40.

De Raedt, L.: 2008b, *Logical and relational learning*, Cognitive technologies, Springer, New York.

De Raedt, L.: 2008c, Representations for mining and learning, *in* D. M. Gabbay and J. Siekmann (eds), *Logical and Relational Learning*, Springer-Verlag, Berlin Heidelberg, Germany, pp. 71–114.

De Raedt, L., Demoen, B., Fierens, D., Gutmann, B., Janssens, G., Kimmig, A., Landwehr, N., Mantadelis, T., Meert, W., Rocha, R. et al.: 2008, Towards digesting the alphabet-soup of statistical relational learning, *NIPS Workshop on Probabilistic Programming.*

Dietterich, T., Domingos, P., Getoor, L., Muggleton, S. and Tadepalli, P.: 2008, Structured machine learning: the next ten years, *Mach Learn* **73**, 3–23.

Dietterich, T. G. and Flann, N. S.: 1997, Explanation-based learning and reinforcement learning: A unified view, *Machine Learning* **28**(2-3), 169–210.

Dobson, P. D. and Doig, A. J.: 2003, Distinguishing enzyme structures from non-enzymes without alignments, *J. Mol. Biol.* **330**(4), 771–783.

Duarte, A. P., Cidade, M. T. and Bordado, J. C.: 2006, Cellulose acetate reverse osmosis membranes: Optimization of the composition, *Journal of Applied Polymer Science* **100**(5), 4052–4058.

Dudek, A. Z., Arodz, T. and Galvez, J.: 2006, Computational methods in developing quantitative structure-activity relationships (QSAR): a review, *Combinatorial chemistry & high throughput screening* **9**(3), 213–228.

Emmerich, M., Deutz, A. and Klinkenberg, J.-W.: 2008, The computation of the expected improvement in dominated hypervolume of pareto front approximations, *Technical Report LIACS-TR 9-2008*, Leiden Institute for Advanced Computer Science, Leiden University, Niels Bohrweg 1, Leiden University, The Netherlands.

Fletcher, R.: 1987, *Practical Methods of Optimization*, John Wiley & Sons, New York.

Floyd, R. W.: 1962, Algorithm 97, shortest path, *Commun. ACM* **5**(6), 345.

Form, N., Burbidge, R., Ramon, J. and Whitaker, J.: 2007, Parameterisation of an acousto-optic programmable dispersive filter for closed-loop learning experiments, *Journal of Modern Optics* **55**(1), 1–13.

Frasconi, P., Costa, F., De Raedt, L. and De Grave, K.: 2011, kLog: a language for logical and relational learning with kernels, *Technical report*, Katholieke Universiteit Leuven, Università degli Studi di Firenze, and Albert-Ludwigs-Universität.

Free, S. M. and Wilson, J. W.: 1964, A mathematical contribution to structure-activity studies, *Journal of Medicinal Chemistry* **7**(4), 395–399.

Garcia-Molina, H., Ullman, J. D. and Widom, J.: 2009, *Database systems: the complete book*, 2nd ed edn, Pearson Prentice Hall, Upper Saddle River, N.J.

Gärtner, T.: 2003, A survey of kernels for structured data, *SIGKDD Explor.* **5**(1), 49–58.

Gärtner, T., Horváth, T., Le, Q. V., Somla, A. J. and Wrobel, S.: 2007, Kernel methods for graphs, *in* D. J. Cook and L. B. Holder (eds), *Mining Graph Data*, John Wiley and Sons, Hoboken, NJ.

Gevers, L.: 2005, *The Development and Application of Improved Solvent-resistant Nanofiltration Membranes*, PhD thesis, Katholieke Universiteit Leuven, Belgium.

Gibbs, M.: 1997, *Bayesian Gaussian Processes for Regression and Classification*, PhD thesis, University of Cambridge.

Gillet, V. J., Willett, P. and Bradshaw, J.: 2003, Similarity searching using reduced graphs, *J. Chem. Inf. Comput. Sci.* **43**(2), 338–345.

Goldberg, D.: 1989, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley.

Gower, J.: 1971, A general coefficient of similarity and some of its properties, *Biometrics* **27**, 857–871.

Gribskov, M. and Robinson, N. L.: 1996, Use of receiver operating characteristic (roc) analysis to evaluate sequence matching, *Comput. Chem.* **20**(1), 25–33.

Gross, J. L. and Yellen, J.: 2003, *Handbook of Graph Theory (Discrete Mathematics and its Applications)*, 1st edn, CRC Press.

Guestrin, C., Krause, A. and Singh, A. P.: 2005, Near-optimal sensor placement in gaussian processes, *Proceedings of the 22nd International Conference on Machine Learning*, pp. 265–272.

Guha, R. et al.: 2007, The Open Babel Package, version 2.1.0. http://openbabel.sourceforge.net/.

Guha, R. et al.: 2010, The Open Babel Package, version 2.2.3. http://openbabel.sourceforge.net/.

Haddada, R., Ferjani, E., Roudesli, M. S. and Deratani, A.: 2004, Properties of cellulose acetate nanofiltration membranes. application to brackish water desalination, *Desalination* **167**, 403–409. Desalination Strategies in South Mediterranean Countries.

Hansch, C., Maloney, P. P., Fujita, T. and Muir, R. M.: 1962, Correlation of biological activity of phenoxyacetic acids with hammett substituent constants and partition coefficients, *Nature* **194**, 178–180.

Haussler, D.: 1999, Convolution kernels on discrete structures, *Technical Report UCS-CRL-99-10*, UC Santa Cruz.

Hert, J., Willett, P., Wilton, D. and Acklin, P.: 2004, Comparison of topological descriptors for similarity-based virtual screening using multiple bioactive reference structures, *Organic & Biomolecular Chemistry* **2**, 3256–3266.

Hofman, J. A. M. H., Beerendonk, E. F., Folmer, H. C. and Kruithof, J. C.: 1997, Removal of pesticides and other micropollutants with cellulose-acetate, polyamide and ultra-low pressure reverse osmosis membranes, *Desalination* **113**(2–3), 209–214. Workshop on Membranes in Drinking Water Production Technical Innovations and Health Aspects.

Hoogenboom, R., Meier, M. A. R. and Schubert, U. S.: 2003, Combinatorial methods, automated synthesis and high-throughput screening in polymer research: Past and present, *Macromolecular Rapid Communications* **24**(1), 15–32.

Horváth, T., Gärtner, T. and Wrobel, S.: 2004, Cyclic pattern kernels for predictive graph mining, *in* W. Kim, R. Kohavi, J. Gehrke and W. DuMouchel (eds), *International Conference on Knowledge Discovery and Data Mining, Proceedings of the Tenth ACM SIGKDD conference (KDD2004)*, ACM, New York, NY.

Horváth, T., Ramon, J. and Wrobel, S.: 2006, Frequent subgraph mining in outerplanar graphs, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, pp. 197–206.

Idris, A., Ismail, A. F., Noordin, M. Y. and Shilton, S. J.: 2002, Optimization of cellulose acetate hollow fiber reverse osmosis membrane production using taguchi method, *Journal of Membrane Science* **205**(1–2), 223–237.

Jaccard, P.: 1901, Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin del la Société Vaudoise des Sciences Naturelles* **37**, 547–579.

Joachims, T.: 1999, Making large-scale SVM learning practical, *in* B. Scholkopf, C. Burges and A. Smola (eds), *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, Cambridge, MA.

Jones, D. R.: 2001, A taxonomy of global optimization methods based on response surfaces, *Journal of Global Optimization* **21**, 345–383.

Jones, D. R. and Schonlau, M.: 1998, Efficient global optimization of expensive black-box functions, *Journal of Global Optimization* **13**(4), 455–492.

Kazius, J., McGuire, R. and Bursi, R.: 2005, Derivation and validation of toxicophores for mutagenicity prediction, *J. Med. Chem.* **48**(1), 312–320.

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Sparkes, A., Whelan, K. E. and Clare, A.: 2009, The Automation of Science, *Science* **324**(5923), 85–89.

King, R., Whelan, K., Jones, F., Reiser, P., Bryant, C., Muggleton, S., Kell, D. and Oliver, S.: 2004, Functional genomic hypothesis generation and experimentation by a robot scientist, *Nature* **427**, 247–252.

Knowles, J.: 2006, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, *Evolutionary Computation, IEEE Transactions on* **10**(1), 50–66.

Knowles, J.: 2009, Closed-loop evolutionary multiobjective optimization, *Computational Intelligence Magazine, IEEE* **4**(3), 77–91.

Knowles, J. and Hughes, E. J.: 2005, Multiobjective optimization on a budget of 250 evaluations, *in* C. A. Coello Coello, A. Hernández Aguirre and E. Zitzler (eds), *Evolutionary Multi-Criterion Optimization*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 176–190.

Kordjamshidi, P., Frasconi, P., Van Otterlo, M., Moens, M.-F. and De Raedt, L.: 2011, Spatial relation extraction using relational learning, *in* A. Tamaddoni-Nezhad (ed.), *Proceedings of the 21st International Conference on Inductive Logic Programming*, Springer-Verlag.

Kramer, S., De Raedt, L. and Helma, C.: 2001, Molecular feature mining in HIV data, *in* D. Lee (ed.), *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, ACM Press, New York, NY, pp. 136–143.

Kushner, H. J.: 1964, A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise, *Journal of Basic Engineering* pp. 97–106.

Lawrence, N., Seeger, M. and Herbrich, R.: 2003, Fast Sparse Gaussian Process Methods: The Informative Vector Machine, *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pp. 609–616.

Lengauer, T., Lemmen, C., Rarey, M. and Zimmermann, M.: 2004, Novel technologies for virtual screening, *Drug Discovery Today* **9**(1), 27–34.

Lipinski, C., Lombardo, F., Dominy, B. and Feeney, P.: 1997, Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings, *Advanced Drug Delivery Reviews* **23**(1–3), 3–25.

Liu, C. and Bai, R.: 2006, Preparing highly porous chitosan/cellulose acetate blend hollow fibers as adsorptive membranes: Effect of polymer concentrations and coagulant compositions, *Journal of Membrane Science* **279**(1–2), 336–346.

Lizotte, D., Wang, T., Bowling, M. and Schuurmans, D.: 2007, Automatic gait optimization with gaussian process regression, *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 944–949.

Lonsdale, H.: 1972, *Theory and practice of reverse osmosis ultrafiltration, in industrial processing with membranes*, Wiley Interscience, NY, chapter VIII, pp. 123–178.

Lukin, O. and Vögtle, F.: 2005, Knotting and threading of molecules: Chemistry and chirality of molecular knots and their assemblies, *Angewandte Chemie International Edition* **44**(10), 1456–1477.

Luks, E. M.: 1982, Isomorphism of graphs of bounded valence can be tested in polynomial time, *Journal of Computer and System Sciences* **25**(1), 42–65.

Malthus, T.: 1798, *An Essay on the Principle of Population*, J. Johnson, London.

McGaughey, G. B., Sheridan, R. P., Bayly, C. I., Culberson, J. C., Kreatsoulas, C., Lindsley, S., Maiorov, V., Truchon, J.-F. and Cornell, W. D.: 2007, Comparison of topological, shape, and docking methods in virtual screening, *Journal of Chemical Information and Modeling* **47**(4), 1504–1519.

McInnes, C.: 2007, Virtual screening strategies in drug discovery, *Current Opinion in Chemical Biology* **11**(5), 494–502.

McKay, M. D., Beckman, R. J. and Conover, W. J.: 1979, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* **21**(2), 239–245.

Menchetti, S., Costa, F. and Frasconi, P.: 2005, Weighted decomposition kernels, *in* L. De Raedt and S. Wrobel (eds), *Proceedings of the 22nd International Conference on Machine Learning*, Vol. 119 of *ACM International Conference Proceeding Series*, ACM, New York, NY, pp. 585–592.

Mercer, J.: 1909, Functions of positive and negative type, and their connection with the theory of integral equations, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **209**, 415–446.

Mezey, P. G.: 2009, QSAR and the ultimate molecular descriptor: the shape of electron density clouds, *J. Math. Chem.* **45**(2), 544–549.

Mitchell, T.: 1997, *Machine Learning*, McGraw-Hill.

Mockus, J.: 1989, *Bayesian Approach to Global Optimization*, Kluwer Academic Publishers.

Morão, A., Escobar, I. C., Pessoa de Amorim, M. T., Lopes, A. and Goncalves, I. C.: 2005, Post synthesis modification of cellulose acetate ultrafiltration membrane for applications in water and wastewater treatment, *Environ. Prog.* **24**(4), 367–382.

Morik, K., Brockhausen, P. and Joachims, T.: 1999, Combining statistical learning with a knowledge-based approach – a case study in intensive care monitoring, *in* I. Bratko and S. Džeroski (eds), *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, Morgan Kaufmann, San Fransisco, CA, pp. 268–277.

Mulder, M.: 1996, *Basic principles of membrane technology*, 2 edn, Kluwer Academic Publishers, The Netherlands.

Munkres, J.: 1957, Algorithms for the assignment and transportation problems, *J. Soc. Indust. and Appl. Math.* **5**(1), 32–38.

Murthy, Z. V. P. and Vengal, J. C.: 2006, Optimization of a reverse osmosis system using genetic algorithm, *Separation Science and Technology* **41**(4), 647–664.

Nishizuka, S., Charboneau, L., Young, L., Major, S., Reinhold, W. C., Waltham, M., Kouros-Mehr, H., Bussey, K. J., Lee, J. K., Espina, V., Peter J. Munson, E., Petricoin, III, L. A. L. and Weinstein, J. N.: 2003, Proteomic profiling of the NCI-60 cancer cell lines using new high-density reverse-phase lysate microarrays, *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 100, pp. 14229–14234.

Rarey, M. and Dixon, S. J.: 1998, Feature trees: A new molecular similarity measure based on tree matching, *J. Comput.-Aided Mol. Des.* **12**, 471–490.

Rasmussen, C. and Williams, C.: 2006, *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, MA, USA.
URL: `http://www.gaussianprocess.org/gpml/chapters/`

Riesen, K. and Bunke, H.: 2009, Reducing the dimensionality of dissimilarity space embedding graph kernels, *Eng. Appl. Artif. Intel.* **22**(1), 48–56.

Rupp, M., Proschak, E. and Scheider, G.: 2007, Kernel approach to molecular similarity based on iterative graph similarity, *J. Chem. Inf. Model.* **47**(6), 2280–2286.

Saigo, H., Nowozin, S., Kadowaki, T., Kudo, T. and Tsuda, K.: 2009, gboost: a mathematical programming approach to graph classification and regression, *Mach. Learn.* **75**, 69–89.

Saljoughi, E., Sadrzadeh, M. and Mohammadi, T.: 2009, Effect of preparation variables on morphology and pure water permeation flux through asymmetric cellulose acetate membranes, *Journal of Membrane Science* **326**(2), 627–634.

Sasena, M. J.: 2002, *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*, PhD thesis, University of Michigan.

Schietgat, L.: 2010, *Graph-based data mining for biological applications*, PhD thesis, Department of Computer Science, K.U.Leuven.
URL: `https://lirias.kuleuven.be/handle/123456789/267094`

Schietgat, L., Costa, F., Ramon, J. and De Raedt, L.: 2009, Maximum common subgraph mining: a fast and effective approach towards feature generation, *in* H. Blockeel, K. Borgwardt and X. Yan (eds), *Proceedings of the 7th International Workshop on Mining and Learning with Graphs*, Vol. 7, Leuven, Belgium, pp. 1–3.

Schietgat, L., Costa, F., Ramon, J. and De Raedt, L.: 2011, Effective feature construction by maximum common subgraph sampling, *Machine Learning* **83**(2), 137–161.

Schietgat, L., Ramon, J., Bruynooghe, M. and Blockeel, H.: 2008, An efficiently computable graph-based metric for the classification of small molecules, *in* J.-F. Boulicaut, M. R. Berthold and T. Horváth (eds), *Proceedings of the 11th International Conference on Discovery Science*, Vol. 5255 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin Heidelberg, Germany, pp. 197–209.

Schneider, G.: 2010, Virtual screening: an endless staircase?, *Nature Reviews Drug Discovery* **9**, 273–276.

Schneider, G., Neidhart, W., Giller, T. and Schmid, G.: 1999, "scaffold-hopping" by topological pharmacophore search: A contribution to virtual screening, *Angewandte Chemie International Edition* .

Schölkopf, B. and Smola, A.: 2002, *Learning with Kernels*, MIT Press, Cambridge, MA.

Schuffenhauer, A., Floersheim, P., Acklin, P. and Jacoby, E.: 2003, Similarity metrics for ligands reflecting the similarity of the target proteins, *J. Chem. Inf. Comput. Sci* **43**(2), 391–405.

Sheridan, R. P. and Kearsley, S. K.: 2002, Why do we need so many chemical similarity search methods?, *Drug Discovery Today* **7**(17), 903–911.

Shervashidze, N. and Borgwardt, K.: 2009, Fast subtree kernels on graphs, *in* Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (eds), *Advances in Neural Information Processing Systems (NIPS) 22*, pp. 1660–1668.

Shoemaker, R.: 2006, The NCI60 human tumour cell line anticancer drug screen, *Nat. Rev. Cancer* **6**, 813–823.

Sorlin, S. and Solnon, C.: 2004, A global constraint for graph isomorphism problems, *in* J.-C. Régin and M. Rueher (eds), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Vol. 3011 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 287–302.

Sourirajan, S. (ed.): 1977, *Reverse Osmosis and synthetic membranes: Theory, Technology, Engineering*, National Research Council Canada.

Sparkes, A., Aubrey, W., Byrne, E., Clare, A., Khan, M. N., Liakata, M., Markham, M., Rowland, J., Soldatova, L. N., Whelan, K. E., Young, M. and King, R. D.: 2010, Towards robot scientists for autonomous scientific discovery, *Automated experimentation* **2**(1).

Srinivasan, A., Muggleton, S. H., King, R. and Sternberg, M.: 1994, Mutagenesis: ILP experiments in a non-determinate biological domain, *Proceedings of the 4th International Workshop on Inductive Logic Programming, volume 237 of GMD-Studien*, pp. 217–232.

Stiefl, N., Watson, I. A., Baumann, K. and Zaliani, A.: 2006, ErG: 2D pharmacophore descriptions for scaffold hopping, *J. Chem. Inf. Model.* **46**(1), 208–220.

Suykens, J. A. K., Van Gestel, T., De Brabander, J., De Moor, B. and Vandewalle, J.: 2002, *Least Squares Support Vector Machines*, World Scientific Publishing, Singapore.

Swamidass, S. J., Azencott, C.-A., Lin, T.-W., Gramajo, H., Tsai, S.-C. and Baldi, P.: 2009, Influence relevance voting: an accurate and interpretable virtual high throughput screening method, *J. Chem. Inf. Model.* **49**, 756–766.

Swamidass, S. J., Chen, J., Bruand, J., Phung, P., Ralaivola, L. and Baldi, P.: 2005, Kernels for Small Molecules and the Prediction of Mutagenicity, Toxicity and Anti-Cancer Activity, *Bioinformatics* **21**(suppl_1), i359–368.

Swinney, D. C. and Anthony, J.: 2011, How were new medicines discovered?, *Nature Reviews Drug Discovery* **10**, 507–519.

Takahashi, Y., Sukekawa, M. and Sasaki, S.-i.: 1992, Automatic identification of molecular similarity using reduced-graph representation of chemical structure, *J. Chem. Inf. Comput. Sci.* **32**(6), 639–643.

Tanimoto, T.: 1957, Internal report, *Technical report*, IBM.

Toivonen, H., Srinivasan, A., King, R. D., Kramer, S. and Helma, C.: 2003, Statistical evaluation of the predictive toxicology challenge 2000-2001, *Bioinformatics* **19**(10), 1183–1193.

Turing, A. M.: 1937, On computable numbers, with an application to the entscheidungsproblem, *Proceedings of the London Mathematical Society* **s2-42**(1), 230–265.

Van Haaren, J. and Van Den Broeck, G.: 2011, Relational learning for football-related predictions, *in* A. Tamaddoni-Nezhad (ed.), *Proceedings of the 21st International Conference on Inductive Logic Programming*, Springer-Verlag.

Vandecasteele, H. and Van Craenenbroeck, E.: 2002, DMax functional group and ring library.

Vandewoestyne, B.: 2008, *Quasi-Monte Carlo Techniques for the Approximation of High-Dimensional Integrals*, PhD thesis, Katholieke Universiteit Leuven.

Vandezande, P., Gevers, L. E. M. and Vankelecom, I. F. J.: 2008, Solvent-resistant nanofiltration: Separating on a molecular level, *Chem. Soc. Rev.* **37**, 365–405.

Vandezande, P., Gevers, L. E. M., Weyens, N. and Vankelecom, I. F. J.: 2009, Compositional optimization of polyimide-based seppi membranes using a genetic algorithm and high-throughput techniques, *Journal of Combinatorial Chemistry* **11**(2), 243–251.

Vandezande, P., Gevers, L. E., Paul, J. S., Vankelecom, I. F. and Jacobs, P. A.: 2005, High throughput screening for rapid development of membranes and membrane processes, *Journal of Membrane Science* **250**(1–2), 305–310.

Vapnik, V.: 1995, *The nature of Statistical Learning Theory*, Springer-Verlag, New York.

Verliefde, A.: 2008, *Rejection of organic micropollutants by high pressure membranes (NF/RO)*, PhD thesis, Delft University, the Netherlands.

Verliefde, A., Cornelissen, E., Amy, G., der Bruggen, B. V. and van Dijk, H.: 2007, Priority organic micropollutants in water sources in Flanders and the Netherlands and assessment of removal possibilities with nanofiltration, *Environmental Pollution* **146**(1), 281–289.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R. and Borgwardt, K. M.: 2010, Graph kernels, *J. Mach. Learn. Res.* **99**, 1201–1242.

Wale, N., Watson, I. and Karypis, G.: 2008, Comparison of descriptor spaces for chemical compound retrieval and classification, *Knowl. Inf. Syst.* **14**, 347–375.

Warmuth, M. K., Liao, J., Rätsch, G., Mathieson, M., Putta, S. and Lemmen, C.: 2003, Active learning with support vector machines in the drug discovery process, *J. Chem. Inf. Comput. Sci.* **43**(2), 667–673.

Warshall, S.: 1962, A theorem on boolean matrices, *J. ACM* **9**(1), 11–12.

Watson, A. and Barnes, R.: 1995, Infill sampling criteria to locate extremes, *Mathematical Geology* **27**, 589–608.

Willett, P.: 2006, Similarity-based virtual screening using 2D fingerprints, *Drug Discovery Today* **11**(23/24), 1046–1051.

Wolf, D., Buyevskaya, O. V. and Baerns, M.: 2000, An evolutionary approach in the combinatorial selection and optimization of catalytic materials, *Applied Catalysis A: General* **200**(1–2), 63–77.

Zhou, M., Liu, H., Kilduff, J. E., Langer, R., Anderson, D. G. and Belfort, G.: 2009, High-throughput membrane surface modification to control nom fouling, *Environmental Science & Technology* **43**(10), 3865–3871.

Publication List

Journal Articles

Cano Odena, A., Spilliers, M., Dedroog, T., De Grave, K., Ramon, J. and Vankelecom, I.: 2011, Optimization of cellulose acetate nanofiltration membranes for micropollutant removal via genetic algorithms and high throughput experimentation, *Journal of Membrane Science* **366**(1-2), 25–32.

De Grave, K. and Costa, F.: 2010b, Molecular graph augmentation with rings and functional groups, *Journal of Chemical Information and Modeling* **50**(9), 1660–1668.

International Conferences and Workshops, Published in Proceedings

Cano Odena, A., Vandezande, P., Cools, I., Vanderschoot, K., De Grave, K., Ramon, J., De Raedt, L. and Vankelecom, I.: 2008, Comparison of multi-parameter optimization strategies for the development of nanofiltration membranes for salt and micropollutants removal, *Proceedings of the International Congress on Membranes and Membrane Processes*, Honolulu, Hawaii.

Costa, F. and De Grave, K.: 2010, Fast neighborhood subgraph pairwise distance kernel, *Proceedings of the 27th International Conference on Machine Learning.*

De Grave, K., Ramon, J. and De Raedt, L.: 2008a, Active learning for high throughput screening, *Proceedings of the Eleventh International*

Conference on Discovery Science, Vol. 5255 of *Lecture Notes in Computer Science*, pp. 185–196. **Received the 2008 Carl Smith Award.**

De Grave, K., Ramon, J. and De Raedt, L.: 2008b, Active learning for primary drug screening, *in* L. Wehenkel, P. Geurts and R. Marée (eds), *Benelearn 08. The Annual Belgian-Dutch Machine Learning Conference*, pp. 55–56.

Posters and Presentations at Miscellaneous Events

De Grave, K. and Costa, F.: 2010a, Augmented molecular graph kernel QSARs, *in* J. Hirst, V. Gillet, J. Goodman, R. Ward, D. Wild and P. Wilett (eds), *Joint Sheffield Conference on Chemoinformatics*, Vol. 5 (poster).

De Grave K., Guns T., Vandekerckhove T.T.M., Landuyt B., Menschaert G., Van Criekinge W., Schoofs L., Luyten W.: 2010, Mining cleavage sites of the mouse peptidome, *in* G. Barton and D. Higgins (eds), *Proceedings of the 9th European Conference on Computational Biology (ECCB10). Sequence Analysis, Alignment and Next Generation Sequencing*, pp. 63 (poster).

De Grave K., Ramon, J., De Raedt, L.: 2008, Active learning for primary drug screens, *at:* Spring Workshop on Mining and Learning, Traben-trarbach, Germany.

De Grave K.: 2007, New assets for life science data and text mining, *at:* FFLF Workshop on Machine Learning, Massembre, Belgium.

De Grave K.: 2008, Active learning for drug lead discovery, *at:* Seminar on Logic and Computation, Oostduinkerke, Belgium.

De Grave K.: 2010, Increasing the generalisation power of graph kernel based QSAR models, *at:* Eve General Meeting, Birmingham, United Kingdom.

Index