Chapter 1

INDUCTIVE QUERIES FOR A DRUG DESIGNING ROBOT SCIENTIST

Ross D. King[1], Amanda Schierz[2], Amanda Clare[1], Jem Rowland[1], Andrew Sparkes[1], Siegfried Nijssen[3], Jan Ramon[3]
[1]Department of Computer Science, Llandinam Building, Aberystwyth University, Aberystwyth, Ceredigion, SY23 3DB, United Kingdom;[2]DEC, Poole House, Bournemouth University, Poole, Dorset, BH12 5BB[3]Departement Computerwetenschappen, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001, Leuven, Belgium.

Abstract: It is increasingly clear that machine learning algorithms need to be integrated in an iterative scientific discovery loop, in which data is queried repeatedly by means of *inductive queries* and where the computer provides guidance to the experiments that are being performed. In this chapter, we summarise several key challenges in achieving this integration of machine learning and data mining algorithms in methods for the discovery of Quantitative Structure Activity Relationships (QSARs). We introduce the concept of a robot scientist, in which all steps of the discovery process are automated; we discuss the representation of molecular data such that knowledge discovery tools can analyse it, and we discuss the adaptation of machine learning and data mining algorithms to guide QSAR experiments.

Key words: Quantitative Structure Activity Relationships, Robot Scientist, Graph Mining, Inductive Logic Programming, Active Learning.

1. Introduction

The problem of learning Quantitative Structure Activity Relationships (QSARs) is an important inductive learning task. It is central to the rational design of new drugs and therefore critical to improvements in medical care. It is also of economic importance to the pharmaceutical industry. The QSAR problem is: given a set of molecules with associated pharmacological activities (e.g. killing cancer cells), find a predictive mapping from structure to activity which enables the design of a new molecule with maximum activity. Due to its importance, the problem has received a lot of attention from academic researchers in data mining and machine learning. In these approaches, a dataset is usually constructed by a chemist by means of experiments in a wet laboratory and machine learners and data miners use

the resulting datasets to illustrate the performance of newly developed predictive algorithms. However, such an approach is divorced from the actual practice of drug design where cycles of QSAR learning and new compound synthesis are typical. Hence, it is necessary that data mining and machine learning algorithms become a more integrated part of the scientific discovery loop. In this loop, algorithms are not only used to find relationships in data, but also provide feedback as to which experiments should be performed and provide scientists interpretable representations of the hypotheses under consideration.

Ultimately, the most ambitious goal one could hope to achieve is the development of a *robot scientist for drug design*, which integrates the entire iterative scientific loop in an automated machine, i.e., the robot not only performs experiments, but also analyses them and proposes new experiments. Robot Scientists have the potential to change the way drug design is done, and enable the rapid adoption of novel machine-learning/data-mining methodologies for QSAR. They however pose particular types of problems, several of which involve machine learning and data mining. These challenges are introduced further in Section The Robot Scientist Eve.

The point of view advocated in this book is that one way to support iterative processes of data analysis, is by turning isolated data mining tools into *inductive querying* systems. In such a system, a run of a data mining algorithm is seen as calculating an answer to a query by a user, whether this user is a human or a computerized system, such as a robot scientist. Compared to traditional data mining algorithms, the distinguishing feature of an inductive querying system is that it provides the user considerably more freedom in formulating alternative mining tasks, often by means of *constraints*. In the context of QSAR, this means that the user is provided with more freedom in how to deal with representations of molecular data, can choose the constraints under which to perform a mining task, and has freedom in how the results of a data mining algorithm are processed.

This chapter summarizes several of the challenges in developing and using inductive querying systems for QSAR. We will discuss in more detail three technical challenges that are particular to iterative drug design: the representation of molecular data, the application of such representations to determine an initial set of compounds for use in experiments, and mechanisms for providing feedback to machines or human scientists performing experiments.

A particular feature of molecular data is that essentially, a molecule is a structure consisting of atoms connected by bonds. Many well-known machine learning and data mining algorithms assume that data is provided in a tabular (attribute-value) form. To be able to learn from molecular data, we

either need strategies for transforming the structural information into a tabular form or we need to develop algorithms that no longer require data in such form. This choice of representation is important both to obtain reasonable predictive accuracy and to make the interpretation of models easier. Furthermore, within an inductive querying context, one may wish to provide users with the flexibility to tweak the representation if needed. These issues of representation will be discussed in Section Representations of Molecular Data in more detail.

An application of the use of one representation is discussed in Section Selecting Compounds for a Drug Screening Library, in which we discuss the selection of compound libraries for a robot scientist. In this application it turns out to be of particular interest to have the flexibility to include background knowledge in the mining process by means of *language bias.* The goal in this application is to determine the library of compounds available to the robot: even though the experiments in a robot scientist are automated, in its initial runs it would not be economical to synthesise compounds from scratch and the use of an existing library is preferable. This selection is however important for the quality of the results and hence a careful selection using data mining and machine learning tools is important.

When using the resulting representation in learning algorithms, the next challenge is how to improve the selection of experiments based on the feedback of these algorithms. The algorithms will predict that some molecules are more active than others. One may choose to exploit this result and perform experiments on predicted active molecules to confirm the hypothesis; one may also choose to explore further and test molecules about which the algorithm is unsure. Finding an appropriate balance between exploration and exploitation is the topic of Section Active learning of this chapter.

2. The Robot Scientist Eve

A Robot Scientist is a physically implemented laboratory automation system that exploits techniques from the field of artificial intelligence to execute cycles of scientific experimentation. A Robot Scientist automatically: originates hypotheses to explain observations, devises experiments to test these hypotheses, physically runs the experiments using laboratory robotics, interprets the results to change the probability that the hypotheses are correct, and then repeats the cycle (Figure 1). We believe that the development of Robot scientists will change the relationship between machine-learning/data-mining and industrial QSAR.

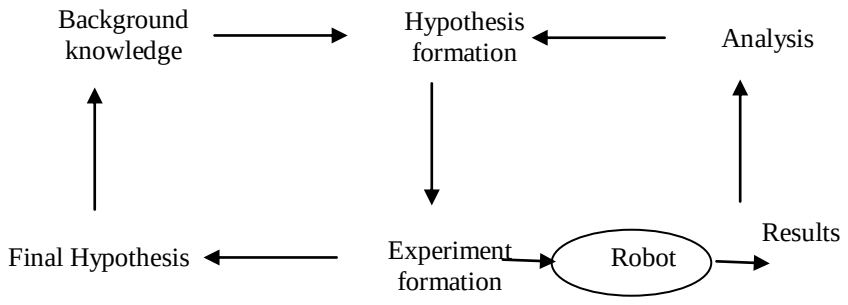

Figure 1. The Robot Scientist hypothesis generation, experimentation, and knowledge formation loop.

In Aberystwyth, we have demonstrated the utility of the Robot Scientist "Adam" which can automate growth experiments in yeast. Adam is the first machine to have autonomously discovered novel scientific knowledge (King et al., 2009). We are now built a new Robot Scientist for chemical genetics and drug design: Eve. This was physically commissioned in the early part of 2009 (see Figure 2). Eve is a prototype system to demonstrate the automation of closed-loop learning in drug-screening and design. Eve's robotic system is capable of moderately high-throughput compound screening (greater than 10,000 compounds per day) and is designed to be flexible enough such that it can be rapidly re-configured to carry out a number of different biological assays.

One goal with Eve is to integrate an automated QSAR approach into the drug-screening process. Eve will monitor the initial mass screening assay results, generate hypotheses about what it considers would be useful compounds to test next based on the QSAR analysis, test these compounds, learn from the results and iteratively feed back the information to more intelligently home in on the best lead compounds.

Eve will help the rapid adoption of novel machine-learning/data-mining methodologies to QSAR in two ways:

- It tightly couples the inductive methodology to the testing and design of new compounds, enabling chemists to step-back and concentrate on the chemical and pharmacological problems rather than the inductive ones.

- It enables inductive methodologies to be tested under industrially realistic conditions.

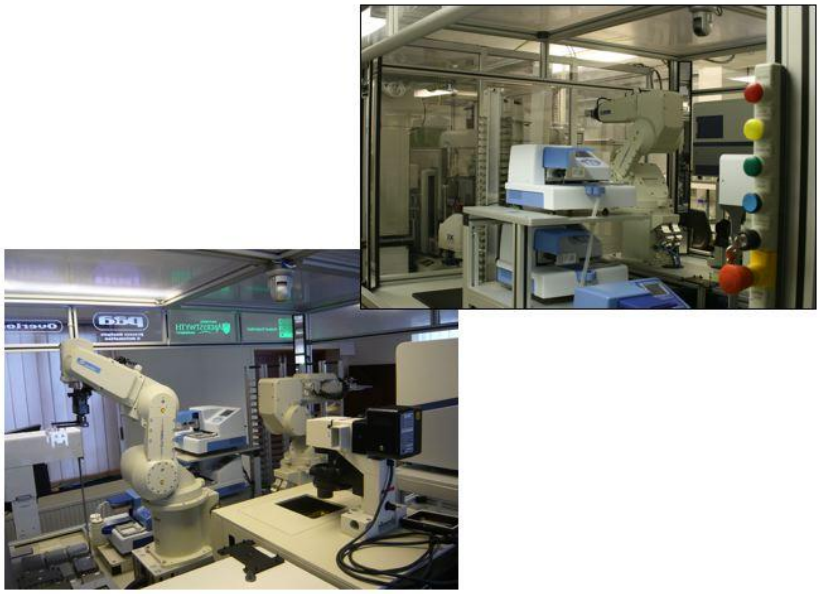

Figure 2. Pictures of Eve

2.1 Eve's Robotics

Eve's robotic system contains various instruments including a number of liquid handlers covering a diverse range of volumes, and so has the ability to prepare and execute a broad variety of assays. One of these liquid handlers uses advanced non-contact acoustic transfer, as used by many large pharmaceutical companies. For observation of assays, the system contains two multi-functional microplate readers. There is also a cellular imager that can be used to collect cell morphological information, for example to see how cells change size and shape over time after the addition of specific compounds.

2.2 Compound Library and Screening

In drug screening, compounds are selected from a "library" (a set of stored compounds) and applied to an "assay" (a test to determine if the compound is active – a "hit"). This is a form of "Baconian" experimentation – what will happen if I execute this action [Med79]. In standard drug screening there is no selection in the ordering of compounds to assay: "Start at the

beginning, go on until you get to the end: then stop" (Mad Hatter, Lewis Carroll). Eve is designed to test an active learning approach to screening.

Eve will initially use an automation-accessible compound library of 14,400 chemical compounds, the Maybridge 'Hit-finder' library (www.maybridge.com). This compound library is cluster-based and was developed specifically to contain a diverse range of compounds. We realise this is not a large compound library - a pharmaceutical company may have many hundreds of thousands or even millions of compounds in its primary screening library. Our aim is to demonstrate the proof-of-principle that incorporating intelligence within the screening process can work better than the current brute-force approach.

2.3 QSAR Learning

In the typical drug design process, after screening has found a set of hits, the next task is to learn a QSAR. This is initially formed from the hits, and then new compounds are acquired (possibly synthesised) and used to test the model. This process is repeated until some particular criterion of success is reached, or too many resources are consumed to make it economical to continue the process. If the QSAR learning process has been successful then a "lead" compound is the result which can then go for pharmacological testing. In machine learning terms such QSAR learning is an example of "active learning" - where statistical/machine learning methods select examples they would like to examine next to optimise learning [DHS01]. In pharmaceutical drug design the *ad hoc* selection of new compounds to test is done by QSAR experts and medicinal chemists based on their collective experience and intuition – there is a tradition of tension between the modellers and the synthetic chemists about what to do next. Eve aims to automate this QSAR learning. Given a set of "hits" from Baconian screening, Eve will switch to QSAR modelling. Eve will employ both standard attribute based, graph based, and ILP based QSAR learning methods to model relationships between chemical structure and assay activity (see below). Little previous work has been done on combining active learning and QSARs, although active learning is becoming an important area of machine learning.

3. Representations of Molecular Data

Many industrial QSAR methods are based around using tuples of attributes or features to describe molecules [HMFM64,Mar78]. An attribute is a proposition which is either true or false about a molecule, for example, solubility in water, the existence of a benzene ring, etc. A list of such propositions is often determined by hand by an expert, and the attributes are

measured or calculated for each molecule before the QSAR analysis starts. This representational approach typically results in a matrix where the examples are rows and the columns are attributes. The procedure of turning molecular structures into tuples of attributes is sometimes called *propositionalization.*

This way of representing molecules has a number of important disadvantages. Chemists think of molecules as structured objects (atom/bond structures, connected molecular groups, 3D structures, etc.). Attribute-value representations no longer express these relationships and hence may be harder to reason about. Furthermore, in most cases some information will be lost in the transformation. How harmful it is to ignore certain information is not always easy to determine in advance.

Another important disadvantage of the attribute-based approach is that is computationally inefficient in terms of space, i.e. to avoid as much loss of information as possible, an exponential number of attributes needs to be created. It is not unusual in chemoinformatics to see molecules described using hundreds if not thousands of attributes.

Within the machine learning and data mining communities, many methods have been proposed to address this problem, which we can categorize along two dimensions. In the first dimension, we can distinguish machine learning and data mining algorithms based on whether they compute *features explicitly,* or operate on the data *directly*, often by having *implicit* feature spaces.

- Methods that compute *explicit* feature spaces are similar to the methods traditionally used in chemoinformatics for computing attribute-value representations: given an input dataset, they compute a table with attribute-values, on which traditional attribute-value machine learning algorithms can be applied to obtain classification or regression models. The main difference with traditional methods in chemoinformatics is that the attributes are not fixed in advance by an expert, but that the data mining algorithm determines from the data which attributes to use. Compared to the traditional methods, this means that the features are chosen much more dynamically; consequently smaller representations can be obtained that still capture the information necessary for effective prediction. The calculation of explicit feature spaces is one of the most common applications of inductive queries, and will hence receive special attention in this chapter.

- Methods that compute *implicit* feature spaces or operate directly on the structured data are more radically different: they do not compute a table with attribute-values, and do not propositionalize the data beforehand. Typically, these methods either directly compute a distance between two molecule

structures, or greedily learn rules from the molecules. In many such models the absence or presence of a feature in the molecule is still used in order to derive a prediction; the main difference is that both during learning and prediction the presence of these features is only determined when really needed; in this sense, these algorithms operate on an *implicit* feature space, in which all features do not need to be calculated on every example, but only on demand as necessary. Popular examples of measures based on implicit feature spaces are *graph kernels*.

- For some methods it can be argued that they operate neither on an implicit nor on an explicit feature space. An example is a largest common substructure distance between molecules. In this case, even though the conceptual feature space consists of substructures, the distance measure is not based on determining the number of common features, but rather on the size of one such feature; this makes it hard to apply most *kernel methods* that assume implicit feature spaces.

The second dimension along which we can categorise methods is the kind of features that are used, whether implicit or explicit:

- *Traditional features* are typically numerical values computed from each molecule by an apriori fixed procedure, such as *structural keys* or *fingerprints,* or features computed through *comparative field analysis.*

- *Graph-based features* are features that check the presence or absence of a sub-graph in a molecule; the features are computed implicitly or explicitly through a data mining or machine learning technique; these techniques are typically referred to as *Graph Mining* techniques.

- *First-order logic features* are features that are represented in a first-order logic formula; the features are computed implicitly or explicitly through a data mining or machine learning technique. These techniques have been studied in the area of *Inductive Logic Programming (ILP).*

We will see in the following sections that these representations can be seen as increasing in complexity; many traditional features are usually easily computed, while applying ILP techniques can demand large computational resources. Graph mining is an attempt to find a middle ground between these two approaches, both from a practical and a theoretical perspective.

3.1 Traditional Representations

The input of the analysis is usually a set of molecules stored in SMILES, SDF or InChi notation. In these files at least the following information about a molecule is described:

- Types of the atoms (such as: is the atom a Carbon, Oxygen, Nitrogen, ...?);
- Types of the bonds between the atoms (such as: is the bond single, double, ...?).

Additionally, these formats support the representation of:

- Charges of atoms (is the atom positively or negatively charged, and how much?);
- Aromaticity of atoms or bond (such as: is an atom part of an aromatic ring?);
- Stereochemistry of bonds (such as: if we have two groups connected by one bond, how can the rotation with respect to each other be categorized?);

Further information is available in some formats, for instance, detailed 3D information of atoms can also be stored in the SDF format. Experimental measurements may also be available, such as the solubility of a molecule in water. The atom-bond information is the minimal set of information available in most databases.

The simplest and oldest approach for propositionalizing the molecular structure is the use of *structural keys*, which means that a finite amount of features are specified beforehand and computed for every molecule in the database. There are many possible structural keys, and it is beyond the scope of this chapter to describe all of these; examples are molecular weight, histograms of atom types, number of hetero-atoms, or more complex features, such as the sum of van der Waals volumes. One particular possibility is to provide an *a priori* list of substructures (OH groups, aromatic rings, ...) and either count their occurrences in a molecule, or use binary features that represent the presence or absence of each a priori specified group.

Another example of a widely used attribute-based method is comparative field analysis (CoMFA) [CPB88]. The electrostatic potential or similar distributions are estimated by placing each molecule in a 3D grid and calculating the interaction between a probe atom at each grid point and the molecule. When the molecules are properly aligned in a common reference frame, each point in space becomes comparable and can be assigned an attribute such that attribute-based learning methods can be used. However, CoMFA fails to provide accurate results when the lack of a common skeleton prevents a reasonable alignment. The need for alignment is a result of the attribute-based description of the problem.

It generally depends on the application which features are most appropriate. Particularly in the case of substructures, it may be undesirable to provide an exhaustive list beforehand by hand. *Fingerprints* were developed to alleviate this problem. Common fingerprints are based on the *graph representation* of molecules: a molecule is then seen as a labelled graph (V,E,λ,Σ) with nodes V and edges E; labels, as defined by a function λ from $V \cup E$ to Σ, represent atom types and bond types. A fingerprint is a binary vector of a priori fixed length n, which is computed as follows:

- All substructures of a certain type occurring in the molecule are enumerated (usually all *paths* up to a certain length);
- A hashing algorithm is used to transform the string of atom and bond labels on each path into an integer number k between 1 and n;
- The kth element of the fingerprint is incremented or set to 1.

The advantage of this method is that one can compute a feature table in a single pass through a database. There is a large variety of substructures that can be used, but in practice paths are only considered, as this simplifies the problems of enumerating substructures and choosing hashing algorithms. An essential property of fingerprints is thus that multiple substructures can be represented by a single feature, and that the meaning of a feature is not always transparent. In the extreme case, one can choose n to be the total number of possible paths up to a certain length; in this case, each feature would correspond to a single substructure. Even though theoretically possible, though, this approach may be undesirable, as one can expect many paths not to occur in a database at all, which leads to useless attributes. Graph mining, as discussed in the next section, proposes a solution to this sparsity problem.

3.2 Graph Mining

The starting point of most graph mining algorithms is the representation of molecules as labelled graphs. In most approaches no additional information is assumed – consequently, the nodes and edges in the graphs are often labelled only with bond and atom types. These graphs can be used to derive explicit features, or can be used directly in machine learning algorithms.

3.2.1 Explicit Features

Explicit features are usually computed through constraint-based mining systems, and will hence be given special attention.

The most basic setting of graph mining is the following.

Definition 1. *Graph Isomorphism*. Graphs $G=(V,E,\lambda,\Sigma)$ and $G'=(V',E',\lambda',\Sigma')$ are called isomorphic if there exists a bijective function f such that: $\forall v \in V$: $\lambda(v)= \lambda'(f(v))$ and $E=\{\{f(v_1),f(v_2)\} \mid \{ v_1, v_2\} \in E'\}$ and $\forall e \in E$: $\lambda(e)= \lambda'(f(e))$.

Definition 2. *Subgraph*. Given a graph $G=(V,E,\lambda,\Sigma)$, graph $G'=(V',E',\lambda',\Sigma')$ is called a subgraph of G iff $V' \subseteq V$ and $E' \subseteq E$ and $\forall v \in V':\lambda'(v')=\lambda(v)$ and $\forall e \in E':\lambda'(e')=\lambda(e)$.

Definition 3. *Subgraph Isomorphism.* Given two graphs $G=(V,E,\lambda,\Sigma)$ and $G'=(V',E',\lambda',\Sigma')$, G is called subgraph isomorphic with G', denoted by $G' \sqsupset G$, iff there is a subgraph G'' of G' to which G is isomorphic.

Definition 4. *Frequent Subgraph Mining*. Given a dataset of graphs $\boldsymbol{D}$, and a graph G, the *frequency* of G in $\boldsymbol{D}$, denoted by $freq(G,\boldsymbol{D})$, is the cardinality of the set $\{G' \in \boldsymbol{D} | G' \sqsupset G\}$. A graph G is *frequent* if $freq(G,D) \geq minsup$, for a predefined threshold *minsup*. The frequent (connected) subgraph mining is the problem of finding a set of frequent (connected) graphs F such that for every possible frequent (connected) graph G there is exactly one graph $G' \in F$ such that G' and G are isomorphic.

We generate as features those subgraphs which are contained in a certain minimum number of examples in the data. In this way, the eventual feature representation of a molecule is dynamically determined depending on the database it occurs in.

There are now many algorithms that address the general frequent subgraph mining problem; examples are AGM [IWM00], FSG [KK01], gSpan [YH02], MoFA [BB02], FFSM [HWP03] and Gaston [NK04]. Some of the early algorithms imposed restrictions on the types of structures considered [KR01, KRH01].

If we set the threshold *minsup* very low, and if the database is large, even if finite, the number of subgraphs can be very large. One can easily find more frequent subgraphs than examples in the database. Consequently, there are two issues with this approach:

- Computational complexity: considering a large amount of subgraphs could require large computational resources.
- Usability: if the number of features is too large, it could be hard to interpret a feature vector.

These two issues are discussed below.

Complexity

Given that the number of frequent subgraphs can be exponential for a database, we cannot expect the computation of frequent subgraphs to proceed in polynomial time. For enumeration problems it is therefore

common to use alternative definitions of complexity. The most important are:

- Enumeration with *polynomial delay*. A set of objects is enumerated with polynomial delay if the time spent between listing every pair of objects is bounded by a polynomial in the size of the input (in our case, the dataset).

- Enumeration with *incremental polynomial time*. Objects are enumerated in incremental polynomial time if the time spent between listing the k and (k+1)th object is bounded by a polynomial in the size of the input and the size of the output till the kth object.

Polynomial delay is more desirable than incremental polynomial time. Can frequent subgraph mining be performed in polynomial time?

Subgraph mining requires two essential capabilities:

- Being able to enumerate a space of graphs such that no two graphs are isomorphic.

- Being able to evaluate subgraph isomorphism to determine which examples in a database contain an enumerated graph.

The theoretical complexity of subgraph mining derives mainly from the fact that the general subgraph isomorphism problem is a well-known NP complete problem, which in practice means that the best known algorithms have exponential complexity. Another complicating issue is that no polynomial algorithm is known to determine if two arbitrary graphs are isomorphic, even though this problem is not known to be NP complete.

However, in practice it is often feasible to compute the frequent subgraphs in molecular databases, as witnessed by the success of the many graph miners mentioned earlier. The main reason for this is that most molecular graphs have properties that make them both theoretically and practically easier to deal with. Types of graphs that have been studied in the literature include;

- Planar graphs, which are graphs that can be drawn on a plane without edges crossing each other [Epp95];

- Outerplanar graphs, which are planar graphs in which there is a Hamilton cycle that walks only around one (outer) face [Lin89];

- Graphs with bounded degree and bounded tree width, which are tree-like graphs[1] in which the degree of every node is bounded by a constant [MT92].

[1] A formal definition is beyond the scope of this chapter.

Graphs of these kinds are common in molecular databases (see Table 1, where we calculated the number of occurrences of certain graph types in the NCI database, a commonly used benchmark for graph mining algorithms).

Graph property	Number
All graphs	250251
Graphs without cycles	21963
Outerplanar graphs	236180
Graphs of tree width 0, 1 or 2	243638
Graphs of tree width 0, 1, 2 or 3	250186

Table 1: The number of graphs with certain properties in the NCI database

No polynomial algorithm is however known for (outer)planar subgraph isomorphism, nor for graphs of bounded tree width without bounded degree and bounded size. However, in recent work we have shown that this does *not* necessarily imply that subgraph mining with polynomial delay or in incremental polynomial time is impossible:

- If subgraph isomorphism can be evaluated in polynomial time for a class of graphs, then we showed that there is an algorithm for solving the frequent subgraph mining algorithm with polynomial delay, hence showing that the graph isomorphism problem can always be solved efficiently in pattern mining [RN08].
- Graphs with bounded tree width can be enumerated in incremental polynomial time, even if no bound on degree is assumed [HR08].
- For the block-and-bridges subgraph isomorphism relation between outerplanar graphs (see next section), we can solve the frequent subgraph mining problem in incremental polynomial time [HRW06].

These results provide a theoretical foundation for efficient graph mining in molecular databases.

Usability

The second problem is that under a frequency threshold, the number of frequent subgraphs is still very large in practice, which affects interpretability and efficiency, and takes away one of the main arguments for using data mining techniques in QSAR.

One can distinguish at least two approaches to limit the number of subgraphs that is considered:

- Modify the subgraph isomorphism relation;

- Apply additional constraints to subgraphs.

We will first look at the reasons for changing the subgraph isomorphism relation.

Changing Isomorphism: Assume we have a molecule containing Pyridine, that is, an aromatic 6-ring in which one atom is a nitrogen. How many subgraphsare contained in this ring only? As it turns out, Pyridine has 2+2+3+3+4+3=17 different subgraphs next to Pyridine itself (ignoring possible edge labels):

```
N  C
C-C  N-C
C-C-C N-C-C  C-N-C
C-C-C-C  N-C-C-C  C-N-C-C
C-C-C-C-C  N-C-C-C-C  C-N-C-C-C  C-C-N-C-C
N-C-C-C-C-C  C-N-C-C-C-C  C-C-N-C-C-C
```

It is possible that each of these subgraphs has a different support; for example, some of these subgraphs also occur in Pyrazine (an aromatic ring with two nitrogens). The support of each of these subgraphs can be hard to interpret without visually inspecting their occurrences in the data. Given the large number of subgraphs, this can be infeasible.

Some publications have argued that the main source of difficulty is that we allow subgraphs which are not rings to be matched with rings, and there are applications in which it could make more sense to treat rings as basic building blocks. This can be formalized by adding additional conditions to subgraph isomorphism matching:

- In [HBB03] one identifies all rings up to length 6 in both the subgraph and the database graph; only a ring is allowed to match with a ring.

- In [HRW06] a *block and bridge preserving* subgraph isomorphism relation is defined, in which bridges in a graph may only be matched with bridges in another graph, and edges in cycles may only be matched with edges in cycles; a *bridge* is an edge that is not part of a cycle.

Comparing both approaches, in [HBB03] only rings up to length 6 or considered; in [HRW06] this limitation is not imposed.

Most subgraph mining algorithms need to be changed significantly to deal with a different definition of subgraph isomorphism. To solve this [HBB03, HRW06] introduce procedures to deal with ring structures.

We are not aware of an experimental comparison between these approaches.

Additional Constraints: The use of constraints is a general methodology to obtain a smaller set of more meaningful subgraphs [**KR01**, **KRH01**]. One can distinguish two types of constraints:

- Structural constraints;
- Data based constraints.

Minimum frequency is one example of a constraint based on data. Many other subgraph types have been proposed based on data constraints:

- Maximally frequent subgraphs, which are subgraphs such that every supergraph in a database is infrequent [KR01, KRH01, HWPY04];
- Closed subgraphs, which are subgraphs such that every supergraph has a different frequency [YH03].
- Correlated subgraphs, which are subgraphs whose occurrences have a significant correlation with a desired target attribute [BZRN06];
- Infrequent subgraphs [KR01, KRH01].

These constraints can be combined. For instance, one can be interested in finding subgraphs that occur frequently in molecules exhibiting a desired property, but not in other molecules.

In practice, these constraints are often not sufficient to obtain small representations. Additional inductive queries can be used to reduce the set of patterns further. A more detailed overview of approaches to obtain smaller sets of patterns is given in another chapter of this book.

An issue of special interest in QSAR applications is which graph types lead to the best results: even though molecules contain cycles, is it really necessary to find cyclic patterns? Experiments investigating this issue can be found in [Nij06, BZRN06,WK06]. The conclusion that may be drawn from these investigations is that in many approaches that use patterns, paths perform equally well as graphs; naïve use of cyclic patterns can even lead to significantly worse results.

3.2.2 Implicit Features & Direct Classification

The alternative to graph mining is to learn classifiers directly on the graph data. The most popular approaches are based on the computation of a *distance* between every pair of graphs in the data. Such distance functions can be used in algorithms that require distance functions, such as *k*-nearest neighbour classification, or support vector machines (SVMs). In SVMs a special type of distance function is needed, the so-called *kernel function*.

One popular type of kernel is the *decomposition kernel*, in which the distance is defined by an implicit feature space. If this implicit feature space

is finite, the kernel value between molecules can in principle be computed by first computing two feature vectors for the pair, and then computing a distance from these feature vectors; the advantage of kernels is that in practice only the (weighted) number of substructures that two particular graphs have in common is computed.

The most commonly used graph kernels are based on the idea of *random walks*: given two molecules, we count the number of walks that both molecules have in common. Note that *walks* differ from *paths* as walks are allowed to visit the same node more than once. If a maximum walk length is given, we could represent two molecules by binary feature vectors with one bit for each possible walk. In practice, though, it is more efficient to scan the two molecules in parallel to make sure we search for common walks. This methodology has further possible advantages. For instance, if we give all walks in graphs a weight which (carefully) shrinks with the length of the walk, a kernel can be defined in which we sum the *infinite* number of such common weighted walks. This number is efficiently computable without explicitly enumerating all walks [GFW03]. Many *kernel* methods have the advantage that they deal easily with possibly infinite representations of structures in a feature space. An early overview of graph kernels can be found in [Gär03], while a more recent overview of walk-based kernels can be found in **Error! Not a valid bookmark self-reference.Error! Not a valid bookmark self-reference.**[Vish09].

Another type of distance function is obtained by computing the *largest common subgraph* of two graphs. The assumption is here that the larger the subgraph is that two molecules have in common, the more similar they are. It is easy to see that this problem is at least as hard as computing subgraph isomorphism. However, the problem may become more easy for the types of graphs identified in the previous section. In [SRBB08] it was shown how to compute the largest common subgraph in polynomial time for outer-planar graphs under the block-and-bridges subgraph relation.

3.2.3 Extended Graph Representations

So far we have considered representations in which nodes correspond to atoms and edges to bonds. This limits the types of knowledge that can be used in the classification. It may be desirable to extend the representation: in some cases it is necessary to classify atom types, e.g. *halogen* (F, Cl, Br, I); to say an atom in an aromatic ring but not specify the atom type; to extend the notion of bond from that of a covalent bond to include non-covalent ones, e.g. hydrogen bonds; etc.

To deal with such issues of ambiguity the common solution is to assume given a *hierarchy* of edge and node labels. In this hierarchy more general

labels, such as ‘halogen’ and ‘hydrogen donor’, are included, as well as the generalization relationships. There are two ways to use these hierarchies:

- We change the subgraph isomorphism operator, such that more general labels are allowed to match with their specialisations [HBB03, Ino04];
- We exploit the fact that in some hierarchies every atom has at most one generalization, by changing the graph representation of the data: we replace the atom type label with the parent label in the hierarchy, and introduce a new node, which is labeled with the original atom type. Optionally, we add additional nodes labeled with other attributes, such as charges [KNK06].

These approaches have mainly been investigated when computing explicit features. An essential problem is then in both approaches the increased number of patterns. Without additional constraints we could find patterns such as `C-Aromatic-C-Aromatic-C` in aromatic rings, that is, patterns in which the labels iterate between specific and general labels. The approaches listed above differ in their approach to avoid or limit such patterns.

3.3 Inductive Logic Programming

In QSAR applications such as toxicity and mutagenicity prediction, where structure is important, Inductive Logic Programming is among the more powerful approaches, and has found solutions not accessible to standard statistical, neural network, or genetic algorithms [DTK98, EK03, KMLS92, KMSS96]. The main distinguishing feature of ILP is that data and models are represented in *first order logic (FOL)*. The classical atom/bond representation in first-order logic is based on the molecular structure hypothesis. Atoms are represented in the form: atom(127,127_1,c,22,0.191), stating that the first atom in compound 127 is a carbon atom of type 22 (aromatic) with a positive charge of 0.191. Similarly, bond(127,127_1,127_6,7) states that there is a type 7 bond (here aromatic) between the first and sixth atom in compound 127. Bonds are represented in a similar fashion.

When only atoms, bonds and their types are represented in FOL facts, the resulting representation is essentially a graph. The main advantage of ILP is the possibility of including additional information, such as charges, and of including background knowledge in the form of computer programs. One example of this is to define a distance measure which enables three-dimensional representations with rules in the form: “A molecule is active if it has a benzene ring and a nitro group separated by a distance of 4 ± 0.5˚A”. The key advantage of this approach to representing three-dimensional

structures is that it does not require an explicit alignment of the molecules. It is also straightforward to include more than one conformation of each compound which allows the consideration of conformation flexibility which is often a major drawback by conventional QSAR/SAR methodologies.

Since chemists often study molecules in terms of molecular groups, the atom/bond representation can be extended with programs that define such high-level chemical concepts. Contrary to propositional algorithms and graph mining, ILP can learn rules which use structural combinations of these multiple types of concepts.

A downside of ILP is the lack of results with respect to efficient theoretical complexity. As shown in the previous section, for many classes of graphs efficient mining algorithms are known. As a result, graph mining is usually efficient, both in theory and in practice. For ILP algorithms no similar theoretical results are available and the algorithms typically require more computational power, both in theory and in practice.

The number of ILP algorithms is very large, and the discussion of this area is beyond the scope of this article. We will limit our discussion here to the relationship between graph mining and ILP algorithms, and approaches that we will need later in this chapter. For a more complete discussion of ILP see [DR08]. An important aspect of ILP algorithms is the background knowledge used. We will conclude this section with a discussion of the details of a library of background knowledge for SAR applications that we recently developed, and is important in allowing users to formulate alternative inductive queries.

3.3.1 Explicit Features

A similar problem as the frequent subgraph mining problem can be formulated in ILP. The data is conceived as a set of definite clauses and facts, for instance:

```
halogen(X,Y) :- atom(X,Y,f,_,_).
halogen(X,Y) :- atom(X,Y,cl,_,_).
...
atom(127,127_1,c,22,0.191).
atom(127,127_2,c,22,0.191).
bond(127,127_1,127_2,single).
```

The database is usually represented as a program in Prolog. The clauses can be thought of as *brackground knowledge*, while the facts describe the

original data. Assume now we are given the following clause, which is not part of the database:

```
f1(X) :-   molecule(X),halogen(X,Y),
           atom(X,Z,c,_,_),bond(X,Y,Z,_).
```

Then for a given constant, for instance 127 in our example, we can evaluate using a Prolog engine whether `f1(127)` is true. If this is the case, we may see `f1` as a feature which describes molecule 127. We may call a clause frequent if it evaluates to true for a sufficient number of examples. The problem of finding frequent clauses is the problem that was addressed in the WARMR algorithm [DDR97].

Definition 4. *Frequent Clause Mining*. Given clause C = `h(X) :- b,` where b is the body of the clause C, and a Prolog database $\boldsymbol{D}$ with constants $\boldsymbol{C}$, the *frequency* of clause C in $\boldsymbol{D}$, denoted by $freq(C,\boldsymbol{D})$, is the cardinality of the set { `c` $\in \boldsymbol{C}$ | $\boldsymbol{D} \wr \{C\}$ ⊢ `h(c)` }; in other words, the number of constants for which we can prove the head of the clause using a Prolog engine, assuming C were added to the data. A clause C is *frequent* if $freq(C,\boldsymbol{D}) \geq minsup$, for a predefined threshold *minsup*. Assume given a language $\boldsymbol{L}$ of clauses. The frequent clause mining is the problem of finding a set of clauses F such that for every possible frequent clause C in $\boldsymbol{L}$ there is exactly one clause $C' \in F$ such that C' and C are equivalent.

It is of interest here to point towards the differences between frequent graph mining and frequent clause mining.

The first practical difference is that most algorithms require an explicit definition of the space of clauses C considered. This space is usually defined in a *bias specification language.* In such a bias specification language, it can be specified for instance that only clauses starting with a `molecule` predicate will be considered, and next to this predicate only `atom` and `bond` predicates may be used. Note that such clauses would essentially represent graphs. The bias specification language can considered a part of the language of an inductive querying system and provides users the possibility to carefully formulate data mining tasks.

The second difference is the use of traditional Prolog engines to evaluate the support of clauses. Prolog engines are based on a technique called *resolution.* There is an important practical difference between resolution and subgraph isomorphism, as typically used in graph mining algorithms. Assume given a clause over only atoms and bonds, for instance,

```
h(X) :-  molecule(X), atom(X,Y,c,_,_),
         bond(X,Y,Z1,_),bond(X,Y,Z2,_)
```

then this clause is equivalent to the following clause:

```
h(X) :- molecule(X), atom(X,Y,c,_,_),bond(X,Y,Z1,_)
```

the reason is that if constants are found for which the second clause succeeds, we can use the same constants to satisfy the first clause, as there is no requirement that Z1 and Z2 are different constants. On the other hand, when using subgraph isomorphism, two atoms in a subgraph may never be matched to the same atom in a molecule.

The use of resolution has important consequences for the procedure that is used for eliminating equivalent clauses. Whereas in graph mining, it is possible to avoid equivalent subgraphs during the search, it can be proved that there are languages of clauses for which this is impossible; the only solution in such cases is to first generate a highly redundant set of clauses, and eliminate duplicates in a post-processing step.

To address this problem, an alternative to resolution was proposed, in which two different variables are no longer allowed to be resolved to the same constant. This approach is known as theta-subsumption under *Object Identity* [DR08].

Similar constraints as proposed in graph mining, can also be applied when mining clauses. However, this has not yet been extensively applied in practice.

3.3.2 Implicit Features & Direct Classification

The alternative to separate feature construction and learning phases is also in ILP to learn a model directly from the data. Contrary to the case of graphs, however, the use of distance functions has only received limited attention in the ILP literature; see [FP08] for a kernel on logical representations of data and [DR09] for a distance based on the *least general generalization* of two sets of literals. The application of these methods on molecular data is yet unexplored; one reason for this is the expected prohibitive performance of these methods, in particular when one wishes to include background knowledge in the *lgg* based methods.

On the other hand, a very common procedure in ILP is to greedily learn a rule-based or tree-based classifier directly from training data; examples of such algorithms include FOIL, Tilde and Progol [DR08]. In graph mining such approaches are rare; the main reason for this is that greedy heuristics are expected to be easily misled when the search proceeds in very “small”, uninformative steps, as common in graph mining when growing fragments bond by bond.

To illustrate one such greedy algorithm, we will discuss the Tilde algorithm here [BD98]. Essentially, Tilde starts from a similar database as WARMR, and evaluates the support of a clause in a similar way as WARMR; however,

as the algorithm is aware of the class labels, it can compute a score for each clause that evaluates how well it separates examples of one or more two classes from each other. For instance, the clause

```
h(X) :- molecule(X), benzene(X,Y)
```

may hold for 15 out of 20 constants identifying positive molecules, and only 15 out of 30 negative molecules; from these numbers we may compute a score, such as information gain:

$$(-0.4 \log 0.4 – 0.6 \log 0.6) - 0.3 (-0.5 \log 0.5 – 0.5 \log 0.5) – 0.7 (-0.25 \log 0.25 – 0.75 \log 0.75)$$

Here the first term denotes the information of the original class distribution (20/50 positives, 30/50 negatives), the second term denotes the information of the examples for which the query succeeds, and the third term denotes the information of the examples for which it fails.

Using such a score, we can compare several alternative clauses. In Tilde clauses are grown greedily, i.e. for a given clause, all possible literals are enumerated that can be added to it, and only the extended clause that achieves the best score is chosen for further extension. If the improvement is too small, the molecules are split in two sets based on whether the clause succeeds. For these two sets of examples, the search for clauses recursively continues. The end result of this procedure constitutes a tree in which internal nodes are labeled with clauses; we can label a leaf by the majority class of the examples ending up in the leaf. This tree can be used directly for classification.

The problem of learning accurate decision trees has been studied extensively, and many techniques, such as pruning, can also be applied on relational decision trees [BD98]. The main downside of algorithms such as Tilde is that the greedy procedure will prevent large carbon-based substructures from being found automatically, as the intermediate steps through which the greedy search would have to go usually do not score exceptionally well on commonly used heuristics. Hence, it is advisable in ILP to specify larger substructures in advance by means of background clauses.

3.3.3 A Library of Chemical Knowledge for Relational QSAR

An important benefit of ILP algorithms is the ability to incorporate background knowledge, for instance, to represent special groups in molecules. The availability of such background knowledge in a data mining system may allow data analysts to query a database from additional perspectives, as will be illustrated in the next section when studying the problem of selecting a library for use in a robot scientist.

To exploit this benefit, it is essential that a comprehensive library of background knowledge is available. We developed a chemical structure background-knowledge-for-learning (Molecular Structure Generator MSG). This consists of a large library of chemical substructures, rings and functional groups, including details of isomers and analogues. This library consists of three main parts (see Appendix 1): a functional group library, a ring library, and a polycycle library. We encoded the standard functional groups have been pre-coded in the library (Appendix 1). The ring library consists of predominantly 3, 4, 5 and 6 length rings. Rings that are identified but do not have specific chemical names are given an *standard* label, for eg, *other_six_ring*. Unnamed rings of up to 15 atoms in length are pre-coded in this way. Appendix 1 shows the specific rings that are in the library. Rings with isomers have been defined individually but they will have a corresponding *parent* predicate held in the library, for eg, isomer_parent(1,3-cyclohexadiene, cyclohexadiene); isomer_parent(1,4-cyclohexadiene, cyclohexadiene). This will mean that inductions may be made over either the specific isomer or for the whole family. The polycycle library consists of predominantly 2 and 3 ring polycycles that have been pre-coded and held in the MSG Prolog library. Polycycles that are not specifically named have been given an *other* label, for e.g. *other_carbon*. All polycycles will be identified regardless if specifically named in the library. Appendix 1 shows the specific polycyles that are in the library. Structures are built up from substructure, for e.g. an anthracene would have facts for 3 benzene rings, 2 fused pair naphthalenes and a polycycle anthracene; an aryl-nitro structure would have facts for a nitro and an aromatic ring. The data have been fully normalised according to Boyce-Codd relational data standards [Cod74]. The example of the representation of the molecule 8-nitroquinoline is shown in Figure 3 and Table 2.

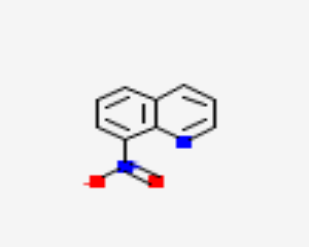

Figure 3. Chemical structure of 8-nitroquinoline

ring_length(2,1,6).	ring_atom(2,2,4).	group_atom(2,5,13).
aromatic_ring(2,1).	ring_atom(2,2,5).	r_atom(2,5,9).
carbon_ring(2,1).	ring_atom(2,2,6).	group(2,6,aryl_nitro).
ring(2,1,benzene).	ring_atom(2,2,7).	part_of_group_structure(2,6,1).
ring_atom(2,1,1).	ring_atom(2,2,8).	part_of_group_structure(2,6,5).
ring_atom(2,1,2).	fused_ring_pair(2,3,1).	count_ring(2,benzene,1).
ring_atom(2,1,3).	fused_ring_pair_share_atom(2,3,8).	count_ring(2,pyridine,1).
ring_length(2,2,6).	polycycle(2,4,quinoline).	count_poly(2,quinoline,1).
n_containing(2,2).	hetero_poly(2,4).	count_group(2,nitro,1).
aromatic_ring(2,2).	poly_no_rings(2,4,2).	count_group(2,aryl_nitro,1).
hetero_ring(2,2).	polycycle_pair(2,4,3).	parent(2,6,nitro).
ring(2,2,pyridine).	group(2,5,nitro).	nextto(2,1,2,fused).
ring_atom(2,2,3).	group_atom(2,5,11).	nextto(2,1,5,bonded).

Table 2. Ground background knowledge generated for 8-nitroquinoline.

4. Selecting Compounds for a Drug Screening Library

This MSG library will be used to generate ILP representations of the compounds that will be screened by the Eve. To test the efficacy of the representation and the method, this library was used to aid the decision-making process for the selection of a compound library to be used with Eve.

The two main criteria for selecting compounds for screening libraries are that they resemble existing approved pharmaceuticals, and that they are structurally diverse. The requirement for a compound in a screening-library to resemble existing pharmaceutically active compounds maximizes the *a priori* probability of an individual compound being active and non-toxic because existing pharmaceutically-active compounds have this property. The requirement for diversity is usually justified by the fact that structurally similar compounds tend to exhibit similar activity - a structurally diverse set of compounds should cover the activity search space and therefore contain fewer redundant compounds [LG03].

Drug-like properties are usually defined in terms of ADME - Absorption, Distribution, Metabolism, and Excretion - and describe the action of the drug within an organism, such as intestinal absorption or blood-brain-barrier penetration. One of the first methods, and still the most popular, to model the absorption property was the “Rule of 5” [LLDF97] which identifies the compounds where the probability of useful oral activity is low. The "rule of 5" states that poor absorption or permeation is more likely when:

1. There are more than 5 Hydrogen-bond donors

2. The Molecular Weight is over 500.
3. The LogP (partition coefficient) is over 5 (or MLogP is over 4.15).
4. There are more than 10 Hydrogen-bond acceptors

The negation of the Lipinski rules are usually used as the main selection criteria for the compounds to include in a screening-library. Though these rules are not definitive, the properties are simple to calculate, and provide a good guideline for drug-likeness.

We have taken an operational approach to determining the drug-likeness of compounds [SK09]. The basic idea is to use machine learning techniques to learn a discrimination function to distinguish between pharmaceutically-active compounds and compounds in screening-libraries. If it is possible to discriminate pharmaceutically-active compounds from compounds in a screening-library then the compounds in the library are considered not drug-like; conversely, if they cannot be discriminated then the compounds are drug-like.

Two compound-screening libraries were chosen for analysis – the target-based NatDiverse collection from Analyticon Discovery (Version 070914) and the diversity-based HitFinder (Version 5) collection from Maybridge. The libraries from these companies are publicly available and this was the main reason for their inclusion in this research. The HitFinder collection includes 14,400 compounds representing the drug-like diversity of the Maybridge Screening Collection (~60,000 compounds). Compounds have generally been selected to be non-reactive and meeting Lipinski's Rule of 5. AnalytiCon Discovery currently offers 13 NatDiverse libraries which are tailor-made synthetic nitrogen-containing compounds. The total number of compounds is 17,402. The approved pharmaceuticals dataset was obtained from the KEGG Drug database and contains 5,294 drugs from the United States and Japan. The data was represented using the Molecular Structure Generator, mentioned above, and the ILP decision tree learner Tilde [BD98], was used to learn the discrimination functions between the set of approved pharmaceuticals and the two compound screening-libraries.

Three tests per dataset were carried out – one based on structural information only, another on quantitative attributes only, and the other based on both structural information and the quantitative attributes. The complete datasets were split into a training and validation set and an independent test set. A ten-fold cross-validation was used for Tilde to learn the decision trees. For each of the three scenarios, the ten-fold cross-validation was carried out with identical training and validation sets. For each scenario, the classification tree that provided the best accuracy when applied to the validation set was applied to the independent test set, see Table 3. The independent test results are good and consistent with validation results. They indicate that the

inclusion of quantitative attributes resulted in increasing the classification accuracy only slightly. The best accuracy was achieved by the decision trees when the data is represented by both structures and properties. These decision trees were represented as a set of Prolog rules and the most accurate rules were selected to build the smallest decision list that had a minimum accuracy of 85%. A complication is the the problem of uneven class distributions (approximately 3:1, screening-library: approved pharmaceuticals)**.**

Testing Dataset	**Accuracy**	**True Negatives**	**True Positives**
HitFinder / App structures only	90%	92%	74%
NAT / App structures only	99%	99%	96%
HitFinder / App properties only	83%	90%	62%
NAT / App properties only	89%	92%	74%
HitFinder / App structures & properties	91%	93%	75%
NAT / App structures & properties	99%	99%	97%

Table 3. Accuracy of the classification trees when applied to the independent test set.

The classification system had more difficulty discriminating approved pharmaceuticals from the diversity-based HitFinder library than the target-based NATDiverse library. However, the ILP method had 91% success in classifying compounds in the HitFinder library and 99% in classifying compounds from the NATDiverse collection when applied to an independent test set. These discrimination functions were expressed in easy to understand rules, are relational in nature and provide useful insights into the design of a successful compound screening-library.

Given a set of rules that can discriminate between drugs and non-drug compounds, the question arises how best to use them in the drug design process. The simplest way to use them would be as filters, and to remove from consideration any compound classed as being non-drug-like. This is what is generally done with the original Lipinski rules - any compounds that satisfy the rules are removed from drug libraries. This approach is non-optimal because such rules are *soft*,as they are probabilistic and can be contravened under some circumstances. However, new data mining research such as multi-target learning research [ZD08] has originated better ways of using prior rules than simply using them as filters. We believe that such approaches could be successfully applied to the drug design problem.

5. Active learning

In many experiment-driven research areas, it is important to select experiments as optimal as possible in order to reduce the amount and the costs of the experiments. This is in particular true for high-throughput screening in the drug discovery process, as thousands of compounds are available for testing. QSAR methods can help to model the results obtained so far. When fit into an active learning strategy, they can be used to predict the expected benefit one can obtain from experiments.

However, in QSAR applications there is an important difference with classical active learning approaches. Usually, one is not interested to get an accurate model for all molecules. It is only important to distinguish the best molecules (and therefore to have an accurate model for the good ones). So instead of active learning where one chooses experiments to improve the global performance of the learned model, in these applications an active optimization approach is desired where one chooses experiments to find the example with the highest target value.

There may be two major reasons why an experiment is interesting. First, one may believe that the molecule being tested has a high probability of being active. In that case, one exploits the available experience to gain more value. Second, the molecule may be dissimilar to the bulk of the molecules tried so far. In that case, the experiment is explorative and one gains new experience from it.

5.1 Selection strategies

Different example selection strategies exist. In geostatistics, they are called infill sampling criteria [Sas02].

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 to greedily improve 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 our model, it translates to *maximum variance* on the predicted value, or $\operatorname{argmax}_t(\operatorname{var}(t))$.

Likely more appropriate for our optimization problem is to select the example that the current model predicts to have the best target value, or $\operatorname{argmax}_t(t)$. We will refer to this as the *maximum predicted* strategy. For

continuous domains, it is well known that it is liable to get stuck in local minima.

A less vulnerable strategy is to always choose the example for which the optimistic guess is maximal. In reinforcement learning, one has shown that with this strategy the regret is bounded (Explicit Explore or Exploit, [KS98]). In that case, the idea is to not (re)sample the example in the database where the expected reward $\bar{t}$ is maximal, but the example where $\bar{t} + b \times \mathrm{var}(\bar{t})$ is maximal. The parameter *b* is the level of optimism. In this paper we do not consider repeated measurements, unlike reinforcement learning, where actions can be reconsidered. This *optimistic* strategy is similar to Cox and John's lower confidence bound criterion [CJ97]. It is obvious that the maximum predicted and maximum variance strategies are special cases of the optimistic strategy, with b=0 and b=∞ respectively. In a continuous domain, this strategy is not guaranteed to find the global optimum because its sampling is not dense [Jon01].

Another strategy is to select the example that has the highest probability of improving the current solution [Kus64]. One can estimate this probability as follows.

Let the current step be *N*, the value of the set of *k* best examples be $\|T_N\|_{best-k}$ and the *k*-th best example be $x_{(k,N)}$ with target value $t_{(k,N)}$. When we query example ,either is smaller than or equal to $t_{(k,N)}$, or t_{N+1} is greater. In the first case, $\|T_{N+1}\|_{best-k} = \|T_N\|_{best-k}$. 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 $P(t_{N+1} > t_{(k,N)})$. We can evaluate this probability by computing the cumulative Gaussian

$$P(t_{N+1} > t_{(k,N)}) = \int N(\bar{t}, \mathrm{var}(t))dt. \quad (1)$$

In agreement with [LWBS07], we call this the *most probable improvement* (MPI) strategy.

Yet another variant is the strategy used in the Efficient Global Optimization (EGO) algorithm [JS98]. EGO selects the example it expects to improve most upon the current best, i.e the one with highest

$$E[\max(0, t - t_{(k,N)})] = \int (t - t_{(k,N)}) N(\bar{t}, \mathrm{var}(t))dt. \quad (2)$$

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

5.2 Effects of properties of experimental equipment.

Most approaches assume an alternation between the algorithm proposing one single experiment and the environment performing one experiment producing a definite answer to the proposed question. After a number of iterations, the algorithm converges then to one optimal solution. However, in practice such a procedure is not always acceptable.

First, in some cases, not all parameters are evaluated during the first stage of experimentation. E.g. in the drug discovery process, active compounds may be rejected at a later stage due to other adverse properties such as toxicity, and therefore one prefers to discover in the first stage several dissimilar candidates instead of one optimal one.

Second, in many applications among which high throughput screening, the equipment can perform several experiments at the same time. E.g. several compounds can be tested on a single plate, or the experiments happen in a pipeline such that several experiments are under way before the result of the first one is known. In such cases, the algorithm has to choose several experiments without knowing the result of all earlier experiments. Therefore, apart from exploitation and exploration, the algorithm also needs diversification.

Third, noise is a common factor in real-world experiments. It means that results are not always exact or trustworthy. Depending on the domain, one may want to perform the same experiment several times, or design different experiments to jointly measure a set of related values.

6. Conclusions

In this chapter we have first introduced the challenges involved in automating the discovery process of new drugs, of which the development of a robot scientist is the arguably the most ambitious. We have provided a more detailed discussion of several of the challenges particular to iterative drug discovery: the representation of molecular data, the use of active learning and the development of libraries that serve as input for the former two tasks.

Even though we made an attempt to provide a reasonably complete summary of the areas and issues involved, the overview in this chapter is far from complete. An important element which is missing from this chapter is an all-encompassing experimental comparison of the representation methods presented (both ILP and graph mining), as well as detailed recommendations with respect to which algorithms to use for which types of data, under which types of constraints or under which type of language bias. Desirable as this

may be, to the best of our knowledge no such comparison is currently available in the literature and most studies have focused on a subset of methods and limited types of data (mostly NCI, see [WK06, BZRN06] for instance). This type of analysis could be a useful topic for further research, for which we hope that this chapter provides some useful hints.

References

[BB02] Christian Borgelt and Michael R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *ICDM*, pages 51–58. IEEE Computer Society, 2002.

[BD98] Hendrik Blockeel, Luc De Raedt: Top-Down Induction of First-Order Logical Decision Trees. Artif. Intell. 101(1-2): 285-297 (1998).

[BDK04] H. Blockeel, S. Dzeroski, B. Kompare, S. Kramer, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. In *Appl. Art. Int. 18*, pages 157–181, 2004.

[BZRN06] Björn Bringmann, Albrecht Zimmermann, Luc De Raedt, and Siegfried Nijssen. Don't be afraid of simpler patterns. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *PKDD*, volume 4213 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2006.

[Cod74] E.F. Codd. Recent Investigations into Relational Data Base Systems. *IBM Research Report RJ1385* (April 23rd, 1974). Republished in Proc. 1974 Congress (Stockholm, Sweden, 1974). New York, N.Y.: North-Holland, 1974.

[CJ97] Dennis D. Cox and Susan John. SDO: a statistical method for global optimization. In *Multidisciplinary design optimization (Hampton, VA, 1995)*, pages 315–329. SIAM, Philadelphia, PA, 1997.

[CPB88] R.D. III Cramer, D.E. Patterson, and Bunce J.D. Comparative Field Analysis (CoMFA). 1. The effect of shape on binding of steroids to carrier proteins. *J. Am. Chem. Soc.* 110: 5959–5967, 1988.

[DTK98] L. Dehaspe, H. Toivonen, and R.D. King. Finding frequent substructures in chemical compounds. In: *The Fourth International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, Ca. 30-36, 1998.

[DDR97] Luc Dehaspe, Luc De Raedt: Mining Association Rules in Multiple Relations. In: *ILP* 1997: 125-132.

[DR08] Luc De Raedt. *Statistical and Relational Learning.* Springer, 2008.

[DR09] Luc De Raedt, Jan Ramon: Deriving distance metrics from generality relations. *Pattern Recognition Letters* 30(3): 187-191 (2009).

[DHS01] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2001.

[EK03] D. Enot and R.D. King. Application of inductive logic programming to structure-based drug design. *Proceedings of the 7th European Conference on*

Principles and Practice of Knowledge Discovery in Databases (PKDD), 2003.

[Epp95] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Symposium on Discrete Algorithms*, pages 632-640, 1995.

[FP08] Paolo Frasconi, Andrea Passerini: Learning with Kernels and Logical Representations. *Probabilistic Inductive Logic Programming*, 2008: 56-91.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.

[GE03] Johann Gasteiger and Thomas Engel. *Chemoinformatics: A Textbook*. Wiley-VCH, 2003.

[GFW03] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *COLT*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2003.

[HMFM64] C. Hansch, P.P. Malony, T. Fujiya, and R.M. Muir, R.M. Correlation of biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients. *Nature* 194, 178-180, 1965.

[HBB03] Heiko Hofer, Christian Borgelt, and Michael R. Berthold. Large scale mining of molecular fragments with wildcards. In Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt, editors, *IDA*, volume 2810 of *Lecture Notes in Computer Science*, pages 376–385. Springer, 2003.

[HCKD04] C. Helma, T. Cramer, S. Kramer, and L. De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. In *Journal of Chemical Information and Computer Systems 44*, pages 1402–1411, 2004.

[HR08] Tamás Horváth and Jan Ramon. Efficient frequent connected subgraph mining in graphs of bounded treewidth. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *ECML/PKDD (1)*, volume 5211 of *Lecture Notes in Computer Science*, pages 520–535. Springer, 2008.

[HRW06] Tamás Horváth, Jan Ramon, and Stefan Wrobel. Frequent subgraph mining in outerplanar graphs. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 197–206. ACM, 2006.

[HWP03] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 549–552. IEEE Press, 2003.

[HWPY04] Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. Spin: mining maximal frequent subgraphs from graph databases. In Won Kim, Ron

Kohavi, Johannes Gehrke, and William DuMouchel, editors, *KDD*, pages 581–586. ACM, 2004.

[Ino04] Akihiro Inokuchi. Mining generalized substructures from a set of labeled graphs. In *ICDM*, pages 415–418. IEEE Computer Society, 2004.

[IWM00] A. Inokuchi, T. Washio, and H. Motoda. An APRIORI-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume 1910 of *Lecture Notes in Artificial Intelligence*, pages 13–23. Springer-Verlag, 2000.

[Jon01] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[JS98] Donald R. Jones and Matthias Schonlau. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, December 1998.

[KK01] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the First IEEE International Conference on Data Mining (ICDM)*, pages 313–320. IEEE Press, 2001.

[KNK06] Jeroen Kazius, Siegfried Nijssen, Joost N. Kok, Thomas Bäck, and Adriaan IJzerman. Substructure mining using elaborate chemical representation. In *Journal of Chemical Information and Modeling 46*, 2006.

[KMLS92] R.D. King, S. Muggleton, R.A Lewis, and M.J.E Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Nat. Acad. Sci. U.S.A.* 89, 11322-11326, 1992.

[KMSS96] R.D. King, S. Muggleton, A. Srinivasan, and M.J.E. Sternberg. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proc. Nat. Acad. Sci. USA* 93, 438-442, 1996.

[KR01] Stefan Kramer and Luc De Raedt. Feature construction with version spaces for biochemical applications. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *ICML*, pages 258–265. Morgan Kaufmann, 2001.

[KRH01] Stefan Kramer, Luc De Raedt, and Christoph Helma. Molecular feature mining in hiv data. In *KDD*, pages 136–143, 2001.

[KS98] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, San Francisco, CA, 1998.

[Kus64] Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, pages 97–106, March 1964.

[LG03] A.R. Leach, and V.J. Gillet. *An Introduction to Chemoinformatics*, Kluwer Academic Publishers, Dordrecht, 2003.

[Lin89] A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science* 63, 295-302, 1989.

[LLDF97] C.A. Lipinski, F. Lombardo, B.W. Dominy, and P. J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings, *Adv. Drug Delivery Rev*., 23(1-3), pp. 3-25, 1997.

[LWBS07] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 944–949, 2007.

[Mar78] Y.C. Martin. *Quantitative Drug Design: A Critical Introduction*, Marcel Dekker, New York, 1978.

[MT92] J. Matousek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. Discrete mathemathics, 108(1-3), 343-364, 1992.

[Med79] P.B. Medewar. *Advice to a Young Scientist*. BasicBooks. 1979.

[Nij06] Siegfried Nijssen. Mining interpretable subgraphs. In *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG)*, 2006.

[NK04] Siegfried Nijssen and Joost N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 2004 International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 647–652. ACM Press, 2004.

[RN08] Jan Ramon and Siegfried Nijssen. Polynomial-delay enumeration of monotonic graph classes. *Journal of Machine Learning Research*, 2009.

[Sas02] M. J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[SK09] A. Schierz, and R.D. King. Drugs and Drug-like compounds: Discriminating Approved Pharmaceuticals from Screening Library Compounds. In *Pattern Recognition in Bioinformatics*, pages 331-343, 2009.

[SRBB08] Leander Schietgat, Jan Ramon, Maurice Bruynooghe, Hendrik Blockeel: An Efficiently Computable Graph-Based Metric for the Classification of Small Molecules. In *Discovery Science* 2008: 197-209.

[Vish09] S. V. N. Vishwanathan, Nicol N. Schraudolph, Imre Risi Kondor, and Karsten M. Borgwardt. Graph Kernels. *Journal of Machine Learning Research*, 2009.

[WK06] Nikil Wale and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *ICDM*, pages 678–689. IEEE Computer Society, 2006.

[YH02] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the Second IEEE International Conference on Data Mining (ICDM)*, pages 721–724. IEEE Press, 2002.

[YH03] Xifeng Yan and Jiawei Han. Closegraph: mining closed frequent graph patterns. In *KDD*, pages 286–295. ACM, 2003.

[ZD08] B. Zenko, and S. Dzeroski. Learning Classification Rules for Multiple Target Attributes. In *PAKDD*, pages 454-465, 2008.

APPENDIX 1

cyclopropane	2,3-dihydropyrrole	benzene
cyclopropene	2,5-dihydropyrrole	pyridine
aziridine	3,4-dihydropyrrole	1,2-dihydropyridine
diaziridine	pyrrolidine	1,4-dihydropyridine
azirine	furan	tetrahydropyridine
diazirine	1,3-dihydrofuran	piperidine
oxirane	2,5-dihydrofuran	4H-pyran
dioxirane	oxolane	2H-pyran
oxirene	1,2-dioxolane	dihydropyran
thiirane	1,3-dioxolane	aromatic_pyran
dithiirane	dioxole	oxane
thiirene	imidazole	thiane
oxathiirane	imidazolidine	dihydrothiopyran
oxaziridine	dihydroimidazole	pyridazine
thiaziridine	pyrazole	1,2-diazinane
dioxathiirane	pyrazoline	1,3-diazinane
cyclobutane	1,2,3-triazole	tetrahydropyridazine
cyclobutene	1,2,4-triazole	pyrimidine
cyclobutadiene	dihydrotriazole	dihydropyrimidine
azetidine	tetrazole	3H-pyrimidine
2,3-dihydroazete	1,3-oxazole	pyrazine
oxetane	1,2-oxazole	tetrahydropyrazine
1,2-dioxetane	dihydrooxazole	piperazine
1,3-dioxetane	1,3,4-oxadiazole	morpholine
thietane	1,2,5-oxadiazole	1,3-oxazinane
1,2-dithietane	1,2,4-oxadiazole	1,2-oxazinane
1,3-dithietane	thiazole	dihydro-1,2-oxazin
cyclopentane	1,3,4-thiadiazole	dihydro-1,3-oxazin
cyclopentene	1,2,5-thiadiazole	1,3-oxazin
cyclopentadiene	1,2,3-thiadiazole	1,3-thiazinane
thiolane	1,2,4-thiadiazole	thiomorpholine
1,2-dithiolane	dihydrothiazole	1,3-dithiane
1,3-dithiolane	thiazolidine	1,4-dithiane
1,2-dithiole	isothiazole	1,4-dioxane
1,3-dithiole	cyclohexane	1,3-dioxane
thiophene	cyclohexene	1,2-dioxane
2,3-dihydrothiophene	1,3-cyclohexadiene	1,4-dioxene
2,5-dihydrothiophene	1,4-cyclohexadiene	dihydrodioxin

pyrrole	triazine cycloheptane

Appendix Table 1: Specific ring structures pre-coded in the MSG library.

benzocyclobutene	acridine	pyrrolizine
benzofuran	perimidine	pyridopyrimidine
indole	beta_carboline	oxanthrene
isoindole	pteridine	chromene
benzothiophene	phenoxazine	isochromene
benzimidazole	phenothiazine	naphthalene
indazole	phenazine	pentalene
benzoxazole	phenanthroline	indene
benzisoxazole	naphthyridine	as-indacene
benzothiazole	carbazole	s-indacene
purine	phthalazine	biphenylene
quinoline	1H-quinolizine	acenaphthylene
isoquinoline	9H,4H- quinolizine	fluorene
quinoxaline	2H-quinolizine	phenalene
quinazoline	indolizine	phenanthrene
cinnoline	pyrrolopyridine	anthracene

Appendix Table 2: Specific polycyclic structures pre-coded in the MSG library.

alkyl_halide	aryl-thioether	methoxy
aryl-halide	carboxylic acid	chain ether
carboxylic-acid halide	ester	aryl ether
hydroxyl	amide	imine
alcohol	other carbonyl	nitro
hetero aryl alcohol	0H-amine	aryl nitro
phenols	1H-amine	nitroso
aldehyde	2H-amine	aromatic nitroso
ketone	ammonium	azo
thiol	aromatic amine	aromatic azo
sulfonic acid	hydroxylamine	aliphatic chain length 5
sulfonyl	phosphoric acid	butyl
sulfone	phosphate	propyl
sulfonamide	phosphonate	ethyl
cyclic thioether	phosphinate	norm methyl
chain thioether	cyclic ether	haloalkane methyl
methylene single	haloalkane methylene	
methylene double	heteroatoms single bonded	
methylene valence		
aliphatic halide		

Table 3: Specific functional groups pre-coded in the library.