Large Classifier Systems in Bio- and Cheminformatics

Jörg S. Wicker

TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Bioinformatik

Large Classifier Systems in Bio- and Cheminformatics

Jörg S. Wicker

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Die Dissertation wurde am 15.07.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 02.09.2013 angenommen.

What's the most you've ever lost on a coin toss?

Anton Chigurh

Abstract

Large classifier systems are machine learning algorithms that use multiple classifiers to improve the prediction of target values in advanced classification tasks. Although learning problems in bio- and cheminformatics commonly provide data in schemes suitable for large classifier systems, they are rarely used in these domains. This thesis introduces two new classifiers incorporating systems of classifiers using Boolean matrix decomposition to handle data in a schema that often occurs in bio- and cheminformatics.

The first approach, called MLC-BMaD (multi-label classification using Boolean matrix decomposition), uses Boolean matrix decomposition to decompose the labels in a multi-label classification task. The decomposed matrices are a compact representation of the information in the labels (first matrix) and the dependencies among the labels (second matrix). The first matrix is used in a further multi-label classification while the second matrix is used to generate the final matrix from the predicted values of the first matrix. MLC-BMaD was evaluated on six standard multi-label data sets, the experiments showed that MLC-BMaD can perform particularly well on data sets with a high number of labels and a small number of instances and can outperform standard multi-label algorithms. Subsequently, MLC-BMaD is extended to a special case of multi-relational learning, by considering the labels not as simple labels, but instances. The algorithm, called ClassFact (Classification factorization), uses both matrices in a multi-label classification. Each label represents a mapping between two instances. Experiments on three data sets from the domain of bioinformatics show that ClassFact can outperform the baseline method, which merges the relations into one, on hard classification tasks.

Furthermore, large classifier systems are used on two cheminformatics data sets, the first one is used to predict the environmental fate of chemicals by predicting biodegradation pathways. The second is a data set from the domain of predictive toxicology. In biodegradation pathway prediction, I extend a knowledge-based system and incorporate a machine learning approach to predict a probability for biotransformation products based on the structure- and knowledge-based predictions of products, which are based on transformation rules. The use of multi-label classification improves the performance of the classifiers and extends the number of transformation rules that can be covered. For the prediction of toxic effects of chemicals, I applied large classifier systems to the ToxCast™ data set, which maps toxic effects to chemicals. As the given toxic effects are not easy to predict due to missing information and a skewed class distribution, I introduce a filtering step in the multi-label classification, which finds labels that are usable in multi-label prediction and does not take the others in the prediction into account. Experiments show that this approach can improve upon the baseline method using binary classification, as well as multi-label approaches using no filtering.

The presented results show that large classifier systems can play a role in future research challenges, especially in bio- and cheminformatics, where data sets frequently consist of more complex structures and data can be rather small in terms of the number of instances compared to other domains.

Zusammenfassung

Große Klassifikationssysteme sind Algorithmen des Maschinellen Lernens, die mehrere Klassifizierer kombinieren, um die Vorhersagen von Zielvariablen in fortgeschrittenen Klassifikationsproblemen zu verbessern. Obwohl die Daten von Lernprobleme in Bio- und Chemieinformatik häufig in einer Form existieren, die große Klassifikationssysteme ermöglichen, werden sie in diesen Domänen nur selten benutzt. Diese Dissertation führt zwei neue Klassifizierer ein, die Boolesche Matrixzerlegung benutzen, um Daten zu verarbeiten, die häufig in Bio- und Chemieinformatik vorkommen.

Der erste Ansatz, MLC-BMaD (Multi-Label Classification using Boolean Matrix Decomposition), benutzt Boolesche Matrixzerlegung, um Labels in einem Multi-Label Klassifikationsproblem zu zerlegen. Die zerlegten Matrizen sind eine kompakte Repräsentation der Informationen in den Labels (erste Matrix) und der Abhängigkeiten zwischen den Labels (zweite Matrix). Die erste Matrix wird in eine weiteren Multi-Label Klassifikation verwendet, während die zweite verwendet wird, um die finale Matrix aus den vorhergesagten Werten der ersten Matrix zu generieren. MLC-BMaD wurde auf sechs Standard Multi-Label Datensätzen evaluiert. Die Experimente zeigen, dass MLC-BMaD besonders gute Vorhersagen trifft auf Datensätzen mit vielen Labels und wenig Instanzen und bessere Vorhersagen trifft als Standard Multi-Label Klassifizierer. Anschließend wird MLC-BMaD erweitert, um einen besonderen Fall des Multi-Relationalen Lernens abzudecken, indem die Labels nicht als einfache Labels, sondern als Instanzen betrachtet werden. Der Algorithmus, genannt ClassFact (Classification factorization) benutzt beide Matrizen in einer Multi-Label Klassifikation. Jedes Label repräsentiert eine Abbildung zwischen zwei Instanzen. Experimente auf drei Datensätzen aus der Bioinformatik zeigen, dass ClassFact auf schwierigen Klassifikationsproblemen bessere Vorhersagen liefert als die Standard Methode, die alle Relationen in eine Relation vereinigt.

Weiterhin werden große Klassifikationssysteme auf zwei Datensätze aus der Chemieinformatik angewandt, der erste Datensatz verwendet große Klassifikationssysteme, um Abbauprodukte in der Natur vorherzusagen. Der zweite Datensatz kommt aus der Domäne der Toxizitätsvorhersage. In der Vorhersage von Abbauprodukten erweitere ich ein wissensbasiertes System mit einem Ansatz des Maschinellen Lernens, um eine Wahrscheinlichkeit für einzelne Abbauprodukte vorherzusagen, die auf Transformationsregeln basieren. Multi-Label Klassifikation kann die Vorhersagen verbessern und die

Anzahl der verwendeten Transformationsregeln vergrößern. Für die Vorhersage von toxischen Effekten von Chemikalien wurden große Klassifikationssysteme auf dem ToxCast™ Datensatz verwendet. Da die toxischen Effekte aufgrund fehlender Informationen und schiefer Klassenverteilungen nicht leicht vorherzusagen sind, wurde ein Filter Schritt in die Multi-Label Klassifikation eingeführt. Dieser findet Label, die in einer Multi-Label Klassifikation benutzt werden können, und ignoriert andere in der Vorhersage. Experimente zeigen, dass dieser Ansatz bessere Vorhersagen liefert als die Standard Methode, die binäre Klassifikation verwendet, sowie Multi-Label Klassifikation ohne Filtern.

Die präsentierten Ergebnisse zeigen, dass große Klassifikationssysteme eine Rolle in zukünftigen Forschungsherausforderungen spielen können, speziell in Bio- und Chemieinformatik, wo Datensätze häufig eine komplexere Struktur besitzen und klein in Bezug auf die Anzahl der Instanzen sein können.

Acknowledgments

First of all, I have to thank Stefan Kramer. I visited his lecture on machine learning during my undergrad studies, where he introduced me to the fascinating area of research, later giving me the chance to work on a project for developing an inductive database. After finishing my undergrad studies, he invited me to join his group; I accepted his offer and continued working in this area. He provided the perfect mix of inspiration, guidance and freedom in my research, making the work in academia a real pleasure for me. He supported me in the best way while graduating, letting me find my own way and interests in research, but providing me with just enough specific directions and ideas.

In the past years of my research I met, visited and collaborated with many wonderful and inspirational people that influenced my work or my life in one way or another. In the following, I would like to thank them individually for their respective contributions.

With Yana Bromberg and Burkhard Rost I had the chance to work on the prediction of the effects of SNPs. Both not only introduced me to a new research area but also provided me with a new perspective on machine learning.

I visited the Eawag institute in Zürich twice and worked with Kathrin Fenner on biodegradation pathway prediction during and beyond the visits at Eawag. She introduced me to the field of biodegradation pathway prediction and the numerous interesting challenges in this area.

In 2009 I visited the Jožef Stefan Institute in Ljubljana for a month. Many thanks go to everyone in Ljubljana for making my visit the great experience it was, especially Sašo Džeroski who invited me and Darko Aleksovski, with whom I had numerous interesting discussions on multi-label classification.

A big part of the last years, I worked on the OpenTox project. In this respect, I want to thank Barry Hardy and all partners from the OpenTox project for the great experience I gained from working on this project.

Bernhard Pfahringer visited us in Munich and shared an office for a few months with me. A result of the visit was a joint publication on Boolean matrix decomposition in multi-label classification.

I would like to thank my colleagues at Technische Universität München and at Johannes Gutenberg-Universität Mainz for their support and discussions. Especially I want to mention Matthias Böck for the countless long and interesting discussions on various topics, including naked mole rats. Michael Geilke joined our group only one and a half years ago, yet he was already a great help with writing this thesis, not only being a reliable reminder of many appointments, but also a pleasant office mate. Marianne Müller and Andreas Karwath both always had an open door for questions and discussions. Timothy Karl in Munich was more than a great administrator, providing me with all the computational power I needed for my experiments and providing easy solutions for many problems. In our group, I worked together with Tobias Girschick and Fabian Buchwald in the OpenTox project. It was a great experience to develop the OpenTox TUM services with them. With Benny Kneissl I shared the excitement of the last period of graduation. Most of them not only were great colleagues and responsible for an inspiring and friendly work environment but also became friends in private live.

In the last years, I worked with many students on numerous projects, most notable are two students from Mainz. I joined Andrey Tyukin for a while on a wild ride, which is his life. The ambitious, thoughtful and brilliant ideas of Andrey's mind were and still are a pleasure to see and it still is a great experience to work with him. The other student, Michael Hedderich, recently got involved in a research project of our group. His reliable work and enthusiasm are a great contribution and hopefully he will continue to work on future projects with us as well.

In my undergrad studies, I had a lot of support from my fellow student Kristina Kessler. Among many other occasions where she helped me, she actually introduced me to a lecture on machine learning, held by an, at that time, unknown professor to me, Stefan Kramer.

Last but not least I want to thank Katerina Tashkova for supporting me while writing the thesis. She not only gave me the strength and support to write it, but also a lot of research drastically progressed while traveling to visit her. I wrote papers, chapters for the thesis and code on long and frequent train trips between Mainz, Munich and Ljubljana. Even though short, the visits and weekends in these cities helped me regain my energy until the next trips, for the thesis to be written, and for the upcoming challenges of life.

Contents

CHAPTER 1

Introduction

In recent years, data from the domain of bio- and cheminformatics have become more and more complex as new technologies allow the integration of previously unknown information and the generation of large data sets. An example for this is the ToxCast™ data set [22, 55, 59, 21, 64, 66], which was the first data set with this large number of toxic effects. It maps around 300 chemicals to approximately 400 toxic effect and was released by the environmental protection agency (EPA) of the USA in 2009. The large number of toxic effects requires the use of complex methods to analyze the data set and extract information to predict these effects for new chemicals.

To approach the complexity of this new generation of data, this thesis introduces methods for large classifier systems in bio- and cheminformatics. Here, classification is meant in the context of machine learning [69], which aims to develop learning algorithms. More specifically, the goal of machine learning is, given a task T and a performance measure P, to develop algorithms which improve with respect to P from experience E. One of the most popular tasks in machine learning is classification. Classification is the task of learning a function from data with a known class, which can be applied to data with an unknown target value. The function predicts the target value or class of the new instance. Task T is the prediction of an unknown class to a new instance, performance measure P is typically a measure calculated from predicting known values of instances, and experience E is a set of examples with a known class value. Classification is a common task in bio- and cheminformatics, e.g., the classification of amino acids in a protein to predict their secondary structure element, or the prediction of toxic effects in animals of given chemical structures.

Traditional classification in machine learning covers the so-called binary classification, i.e., the task of predicting a binary class for a given instance. Either one of the two classes can be true for an instance, never none, or both at the same time. While this covers a wide range of real-world problems, many real-world problems cannot be targeted in this way. The first adaptation of classification is to handle multi-class problems [44]. This is the simple extension from a binary class to a non-binary case. An example is assigned to one of multiple classes.

A special case of multi-class classification is multi-label classification [97]. In multi-label classification, the classes (called labels) are non-exclusive, i.e., an example can be assigned to multiple labels at once. Additionally, the labels may depend on each other. The goal of multi-label classification is to exploit the dependencies of the labels to improve the overall prediction. The default case for this kind of data is to consider each label as independent class and learn independent models for each label. This approach does not take the dependencies into account. Nevertheless, if a data set provides enough information per label, i.e., the data set consists of a large number of instances, the dependencies might not be required to learn well performing models. The dependencies can improve the performance of a model if there are only few instances. The goal of multi-label classification is to find algorithms that can produce models that exploit the dependencies in the labels and perform well given data sets with few instances and a large number of labels with high dependencies among them[1].

1.1 Motivation

One of the main types of multi-label classifiers are transformation-based algorithms, that transform the data set into one or more single-label data sets and apply single-label classification algorithms on them [93, 77, 16, 47]. As multiple data sets are generated, sometimes up to one data set per label, the number of classifiers required to be learned can be rather large. Other approaches adapt existing algorithms to predict multiple labels at once. Nevertheless, systems of classifiers tend to perform better as they give the flexibility to choose a base classifier suitable to the problem, while adapted classification algorithms use a base classifier that can be chosen and optimized to perform best on the given data set. Although the field of multi-label classification became more and more popular in recent years, there exist only few algorithms which perform particularly well using data sets with a large number of labels and few instances. One major problem is

1 The number of instances in a multi-label data sets is usually below 500, e.g., the ToxCast™ (see Chapter 5) data set consists of 309 instances. The number of classes can be several thousands, e.g., the EUR-Lex data set [67].

the amount of computing resources, which can increase strongly the bigger the data set becomes. On the other hand, multi-label algorithms might not be necessary if the data set is too small in terms of labels. Nevertheless, transformation-based algorithms are the most common example of a large classifier system in machine learning.

In this thesis, I will present a multi-label classifier that is based on Boolean matrix decomposition (BMD). The classifier, called MLC-BMaD (Multi-label classification using Boolean matrix decomposition), transforms the label space using BMD. This step makes it perform well on data which is large in terms of labels: The learning problem is reduced by the BMD. Another benefit is the extraction of dependencies by the BMD, that are later used to predict the labels via a Boolean multiplication.

MLC-BMaD can be further extended to handle more complex types of data in the form of multi-relational classification [25], where data are stored in multiple relations. The relations are linked to each other, and algorithms incorporate the knowledge of linked relations when learning models on this data. A special, and in real-world data rather common, type of multi-relational classification involves data stored in three relations. Two relations describe sets of instances with certain features, similar to the simplest case in machine learning, when only one relation is used to describe a data set with a set of features. The third relation stores the class information of combinations of two instances, each from one of the two sets.

The most widely used approach to handle this kind of data is to join the data into one relation, enabling the use of standard machine learning algorithms. However, as information is stored redundantly, this approach generates large relations that can be too big to handle with standard machine learning algorithms. In this respect, this thesis proposes an algorithm based on MLC-BMaD, which uses BMD to extract two multi-label classification problems from this kind of data. Classes are predicted using Boolean multiplication of multi-label predictions.

1.2 Applications

While machine learning became a popular technique for a wide range of problems, large classifier systems are still rarely used in potential applications. This might be caused by the relatively young age of large classifier systems, nevertheless, there are many potential areas such as bio- and cheminformatics, which would benefit from the use of large classifier systems.

In cheminformatics, machine learning algorithms are used for many applications, e.g., drug design or toxicity prediction. However, the used algorithms tend to be rather simple. Most of the time, basic classification algorithms are used for prediction. While the field already benefits a lot from these algorithms, more sophisticated approaches can further improve the predictive performance.

1.2.1 Biodegradation Pathway Prediction

One example for the possible benefits of large classifier systems for cheminformatics is biodegradation pathway prediction [43, 27, 36]. Machine learning is not used strongly in this field. The goal is to predict the environmental fate of chemicals. All chemicals which are produced in any industry sooner or later end up in the environment. The compounds are degraded in the environment into new compounds. This process can be repeated until the structure is degraded to a point where it will not degrade anymore. The task is to develop systems which can predict the next step in the degradation process. From this, the complete degradation pathway can be predicted. Biodegradation pathway prediction is an important task for environmental scientists and the chemical industry. Compounds can be degraded to many other compounds. Some of these compounds can be toxic and thus harmful for the environment.

At the moment, expert-based systems are mostly used to predict the biodegradation pathways. Nevertheless, these systems tend to over-predict degradation products, i.e., all theoretically possible products are predicted, no matter if they occur in the environment or not. Hence, manual checks are still required on the output of these systems.

As large data sets are still rare in this field, machine learning models can benefit from large classifier systems. Multi-label classification seems to be beneficial, as there are many potential products of the degradation, i.e., classes and only few instances, as experiments are expensive.

1.2.2 Predictive Toxicology

Another example in cheminformatics is predictive toxicology [41]. The task is to learn models that predict toxic effects of chemicals in animals or humans. This is a major field of cheminformatics with a huge interest of the chemical industry, as it is required to perform a large number of experiments that examine the potential environmental and health threats of any chemical. This usually requires animal experiments to be carried out, which is expensive and furthermore ethically controversial.

Predictive toxicology can avoid a large number of animal experiments by predicting the outcome of the experiment by models learned on given data. The training input is usually a data set with chemicals and one target value representing the toxic effect.

The prediction is a toxic effect of a new chemical. One of the main problems of this field is the small size of the data sets. Animal experiments are expensive and hence only few chemicals can be tested. However, as multiple toxic effects can be observed at once, the data sets often consist of multiple potential target values (toxic effects). Thus, multi-label classification can be beneficial for this kind of data. There clearly exist dependencies among the target values, certain effects can influence others. Most models in predictive toxicology only predict one target value at a time and therefore use standard machine learning approaches.

The data sets for both biodegradation pathway prediction and predictive toxicology consist of compounds. The features of the compounds are descriptors of the compounds. The models are learned using QSAR (quantitative structure-activity relationship) methods [38]. The basis of QSAR is the assumption that the structure of a compound is responsible for the effects of that compound in organisms. Hence, similarity between two compounds on the structural level is assumed to lead to similar effects of a compound in an organism.

1.3 Contributions

There are six main contributions presented in this thesis. First of all, we introduce Boolean matrix decomposition (BMD) to classification. BMD has existed for several years, yet there was no major application of it in classification. Section 3.1.2 introduces a multi-label algorithm which uses BMD to extract dependencies between labels and use the product of a learned vector and one of the decomposed matrices to predict new labels for new instances. This approach seems natural for this kind of problem and experiments prove that the use of BMD is beneficial for multi-label classification.

This classifier using BMD, called MLC-BMaD (Multi-Label Classification using Boolean Matrix Decomposition [106], see Section 3.1.2), is the next contribution. The main goal of multi-label classification is to find methods which exploit the dependencies between the labels to predict new label sets. The experiments show that MLC-BMaD is particular good in the case when dependencies between the labels are more important; on data sets with a high number of labels and a low number of instances. The decomposition step is fast and consumes low resources, hence the classification is faster and needs less memory compared to other multi-label algorithms as fewer models need to be learned. Additionally, this algorithm has a higher classification performance given large data sets.

An extension to MLC-BMaD, named ClassFact (Classification Factorization, see Section 3.2.1), introduces a new classification schema. Although the type of problem it covers is rather common, there exist only few approaches which handle this schema. The default case is to transform it to the propositional form and apply traditional machine learning algorithms to the data set. Yet, this approach loses a lot of information and can consume a lot of resources. The newly introduced schema uses BMD to classify pairs of instances. Experiments show that, using ClassFact, classification tasks can benefit from this schema.

In Chapter 4, we show an application of large classifier systems in biodegradation pathway prediction [104, 105]. We introduce a new approach for biodegradation pathway prediction that combines a knowledge-based system with machine learning, and hence uses the best features of both fields, namely the expert knowledge, which is given by biotransformation rules and known biodegradation products, and assists that knowledge via a pruning of the pathways (generated by these rules) using large classifier systems. The approach not only improves the prediction of biodegradation products, but enables also the prediction of larger pathways: More precisely, products that are incorrectly predicted by expert knowledge based transformation rules are discarded using machine learning. Hence, the correct paths can be followed for a larger number of degradation steps.

Chapter 5 applies multi-label classification algorithms to a predictive toxicology data set, the ToxCast™ data set by the Environmental Protection Agency (EPA). The data set maps chemicals to multiple toxic effects in mouse, rat, and rabbit. For each combination of chemical and toxic effect, the data set tells if this effect occurs when the chemical is administered to the corresponding animal. The data set is released in phases, with each phase covering a broader range of chemicals, Phase I of the data set has been published already in 2009. The main goal of phase I is to evaluate the data set and identify potential targets for chemicals in the following phases. In the chapter, we show that, using multi-label classification, dependencies between toxic effects can be exploited. By introducing a filtering approach that uses multi-label classification, groups of toxic effects are found, which can be predicted well together. These groups give indications on potential effects which should be examined in more detail in future releases of the data set.

Another contribution in Chapter 5 is the introduction of multi-label classification to the field of predictive toxicology. Multi-label classification is particularly useful when the known data is big in terms of labels, and small in terms of examples. Additionally, there need to be dependencies between the labels. This is commonly the case in predictive toxicology. Chemicals are tested on a broad range of toxic effects, resulting in many target values which depend on each other, as the toxic effects usually concern similar organs, like liver, brain, or kidney, and are tested on multiple closely related animals like mouse and rat. On the other hand, chemicals are tested in batches, and as the animal experiments are expensive and complex, only a limited number of chemicals are usually tested. Nevertheless, multi-label classification is rarely used in predictive toxicology. So far, only one presentation of experiments using basic multi-label approaches on ToxCast™ exists, yet this approach was never officially published. In this chapter, we extensively use multi-label classification techniques to examine and learn models on the ToxCast™ data set. The experimental results show that predictive toxicology can indeed benefit from the use of multi-label classification.

1.4 Organization of the Thesis

Chapter 2 gives an overview of the state of the art of large classifier systems and previous applications of large classifier systems in biodegradation pathway prediction and predictive toxicology. The main focus is on one of the most common types of large classifier systems, multi-label classification. The notation used in this thesis is introduced, and standard multi-label classifiers are described. Next, I give an introduction to multi-relational learning. The focus lies on a special case of multi-relational learning, where data are given in three relations. Finally, we examine the use of large classifier systems in biodegradation pathway prediction and predictive toxicology.

Subsequently, Chapter 3 covers large classifier systems, i.e., two new algorithms using large classifier systems in multi-label classification and multi-relational learning are introduced. The first classifier, called MLC-BMaD (Multi-Label Classification using Boolean Matrix Decomposition), uses Boolean matrix decomposition to extract dependencies between labels and exploit the dependencies for multi-label classification. The second classifier, ClassFact (Classification Factorization), covers a special case of multi-relational classification. It also uses Boolean matrix decomposition to factorize a central class relation. The decomposition is then used to learn models which predict the classes of combinations of instances.

Next, in Chapter 4 we apply large classifier systems to a problem from the life sciences: The prediction of biodegradation pathways and products. Given a chemical structure, the task is to predict the environmental fate. As a structure can be degraded into many possible other structures, systems of classifiers are learned on biotransformation rules. Besides basic approaches using basic machine learning classifiers, we introduce multi-label classifiers to this task. Using this, we can improve the classification performance.

Another application is introduced in Chapter 5: The prediction of toxic effects of chemical structures. As an application data set, we apply large classifier systems to the ToxCast™ data set as this data set has a structure supporting the use of systems of classifiers: A high number of target values combined with a low number of instances. Due to a high noise level caused by low quality target values, we filter the data set and apply classification to only a subset of the target values which comply to certain quality measures. The approach not only shows that the data set can be used for generating predictive models, but also gives indications of high-potential targets. This result could be used for a further extension of the data set by performing new lab experiments.

Finally, in Chapter 6, I give a conclusion of the work presented in this thesis, and further discuss future research on large classifier systems and their applications to bio- and cheminformatics.

CHAPTER 2

Background

In this chapter, I will present background on the work presented in this thesis. Since the main part focuses on multi-label classification, the first part of this chapter will give an introduction to multi-label classification. It gives the notation used throughout the thesis, evaluation measures used in experiments, and an overview of the most important algorithms in this area. Subsequently, I give an introduction to multi-relational learning. Last, I will give an overview of the two application domains used in this thesis, biodegradation pathway prediction and predictive toxicology.

2.1 Multi-Label Classification

The traditional task in machine learning is to predict one single class for one instance. The models are learned on a data set with known class labels. The task in multi-label classification is to predict not one, but a set of target values, called labels, at once. These labels are interdependent and several can be assigned to one instance. The challenge is to use these dependencies to improve the performance. To achieve this, there exist two categories of classifiers. Transformation-based approaches transform the data set into one or more data sets with only one label and apply standard machine learning algorithms to the data set. The second approach is to modify standard algorithms to be capable of predicting a set of labels instead of one label only. Most of the algorithms introduced here not only predict a set of labels, but also a ranking over the labels, in most cases generated by ordering the labels by the predicted probability.

Especially in transformation-based approaches, it is common to use large systems of classifiers to predict the labels. The most simple one, binary relevance (BR, see Section 2.1.4.1), trains one single model per label. Ensembles of Classifier Chains (ECC, see

Section 2.1.4.3) similarly train one model per label, but use previously used labels as additional input features. In both cases, one classifier is needed per label. Typical multi-label data sets tend to have a large number of labels[2]. Hence, as BR and ECC have to learn one model per label, they generate models with several thousands of classifiers. Such large systems are rarely used for other machine learning tasks.

In the following section, I will give an overview of standard multi-label algorithms with the focus on algorithms using large classifier systems. However, there are also multi-label classifiers not using multiple classifiers. In general, they are adaptations of standard machine learning algorithms (e.g. AdaBoost.MH [83], Rank-SVM [26], Multi-label C4.5 [12], neural networks using BP-MLL [108], or ML-kNN [109]). These methods adapt a standard machine learning algorithm such that they predict multiple target values at once. Strictly speaking, this is not a system of classifiers, but a single classifier.

The field of multi-label classification gained large popularity in recent years [99, 100]. As it is not possible to cover the whole field, I focus on classifiers that are related to the work presented in this thesis.

2.1.1 Notation

The notation used for multi-label classification used in this thesis is presented in Figure 2.1. Similar to traditional classification, features vectors are denoted with $x_i \in \mathcal{X}$. As there exist multiple target values, the labels of an instance are given as label set $Y_i \subseteq \mathcal{Y}$. $\mathcal{Y} = \{\lambda_1,...,\lambda_q\}$ is the set of output labels, each λ_i is one label in one instance. The result of a multi-label classification is a bi-partition into positive labels P_x and negative labels N_x. The number of labels is given by q, the number of instances by m. Furthermore, TP denotes the number of correctly predicted positive labels, TN the number of correctly predicted negative labels, FP the number of labels incorrectly predicted as positive, and FN the number of labels incorrectly predicted as negative.

2.1.2 Data Set Statistics

Multi-label classification is the task of predicting multiple interdependent target values at once. While a data set can have multiple target values, it is not necessarily a good multi-label data set, as there might be no dependencies between the target values. Although data might be identified as multi-label data sets intuitively, a measure to calculate the

2 More than 100 labels is rather common [54, 102, 24, 96], even data sets with several thousands of labels exist [67].

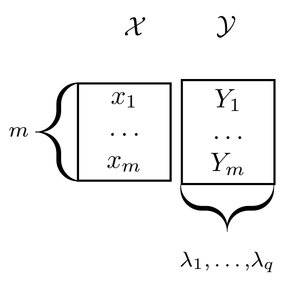


Figure 2.1: Notation used in this section. $\mathcal{Y} = \{\lambda_1, ..., \lambda_q\}$ is the set of output labels. $\mathcal{S} = \{(x_1, Y_1), (x_2, Y_2),...,(x_m, Y_m)\}$ with the feature vector $x_i \in \mathcal{X}$ and the label set $Y_i \subseteq \mathcal{Y}$ gives the instances of a multi-label data set. The bi-partition of the labels is given by the set of predicted labels P_x and not predicted labels N_x.

multi-label quality of a data set does not exist yet. There exist three main statistics for multi-label data sets [97]. *Label cardinality* is the average number of labels per instance:

$$\textit{Label cardinality} = \frac{1}{m} \sum_{i=1}^{m} |Y_i| \tag{2.1}$$

Label cardinality counts the number of labels, and hence gives a measure of the number of labels co-occurring in the data set. However, this might not be a good measure for every data set. If the cardinality is high in a data set with a large number of labels, the labels might co-occur in groups. The groups might be unique in single instances. If there is only small or no overlap among the groups, the labels could be independent with a high *label cardinality.*

Label density is a measure which takes the number of labels into account. *Label density* is the *label cardinality* averaged over the number of labels:

$$\textit{Label density} = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i|}{q} \tag{2.2}$$

Finally, the last statistic used in multi-label classification is the *number of distinct label combinations* in a data set. It should always be considered together with the number of instances in the data set. This measure is particular important for algorithms which incorporate subsets of labels in the learning process, or transform the data set and predict label combinations as classes.

2.1.3 Evaluation Measures

Multi-label classification requires modified measures to handle multiple classifiers and their predictions. The measures can be assigned to certain groups. Based on calculations on the data set, *example-based measures* are averaged over all instances in the test set, while *label-based measures* are averaged over all labels. Based on the prediction of the model, *binary measures* evaluate the prediction of each label, while *ranking measures* consider the ranking of the labels. If the learner returns a probability for the labels, certain measures based on class probability estimates, such as *area under the ROC curve*, can be used as well.

Accuracy is an example-based measure. It gives the fraction of correctly positive predicted labels over the total number of all positive or predicted positive labels:

$$Accuracy = \frac{1}{m}\sum_{i=1}^{m}\frac{|P_i \cap Y_i|}{|P_i \cup Y_i|} \tag{2.3}$$

Coverage counts the steps needed to walk down the list of ranked labels until all correct positive labels are reached. The rank r_i is the position in the ranking produced by multi-label classifiers.

$$Coverage = \frac{1}{m}\sum_{i=1}^{m}\max_{\lambda \in Y_i} r_i(\lambda) - 1 \tag{2.4}$$

Hamming Loss is the percentage of labels which are misclassified. Δ denotes the XOR operator:

$$Hamming\, Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{|P_i \Delta Y_i|}{q} \tag{2.5}$$

One Error is a ranking-based measure. It gives the number how many times the top ranked label is not in the set of positive labels.

$$One\, Error = \frac{1}{m}\sum_{i=1}^{m} I(\operatorname*{argmin}_{\lambda \in \mathcal{Y}} r_i(\lambda) \notin \mathcal{Y}_i) \tag{2.6}$$

where $I(true) = 1$ and $I(false) = 0$.

Ranking Loss is the average number of disordered label pairs per instance.

$$Ranking\ Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{Y}_i||\hat{\mathcal{Y}}_i|}|\{\lambda_a,\lambda_b) : r_i(\lambda_a) > r_i(\lambda_b),(\lambda_a,\lambda_b) \in \mathcal{Y}_i \times \hat{\mathcal{Y}}_i\}| \quad (2.7)$$

Precision is the fraction of correctly predicted labels over all predicted labels:

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|P_i \cap Y_i|}{P_i} \quad (2.8)$$

Recall is the fraction of correctly predicted labels over all positive labels:

$$Recall = \frac{1}{m}\sum_{i=1}^{m}\frac{|P_i \cap Y_i|}{Y_i} \quad (2.9)$$

F1 Measure is the averaged measure of precision and recall combined:

$$F1\ Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.10)$$

Subset Accuracy is the fraction of correctly predicted labels over all positive labels.

$$Subset\ Accuracy = \frac{1}{m}\sum_{i=1}^{m} I(P_i = Y_i), \quad (2.11)$$

Micro Averaged Measures average the predictions on all labels and instances. With *B* being any binary performance measure, *TP* being true positives, *FP* being false positives, *TN* being true negatives, and *FN* being false negatives, it is defined as:

$$B_{micro} = B(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j) \quad (2.12)$$

2.1.4 Transformation-Based Approaches

Transformation-based approaches transform the problem into one or more problems and approach them by using single-label classifiers or multi-class classifiers. Most algorithms in this category transform the data set into single-label problems, e.g., binary relevance, 2BR, multi-label stacking or (ensembles of) classifier chains. Label power set and ensembles of pruned sets transform the data set to enable the use of a multi-class classifier.

The advantage of these algorithms is their flexibility. While in most cases they exploit the dependencies between the labels, they also provide the possibility to choose from a wide range of base classifiers[3]. Hence, the classifiers are very flexible and can adapt to many data sets. Multi-label classifiers adapting existing algorithms (see Section 2.1.5), on the other hand, are restricted to the classifier they adapted, and hence might not be applicable to every data set.

2.1.4.1 Binary Relevance

Binary relevance is a popular example for a problem transformation algorithm. The multi-label classification is divided into multiple single-label classifications (see Algorithm 2.1). The data set is split into q data sets. Each single data set is assigned one label λ_i as class, which results in one single-label problem per label. Single-label classifiers are learned for each data set. To classify a new instance, each single label classifier is applied to the instance, the resulting label set is the combination of all single classifications (see Algorithm 2.2).

Algorithm 2.1: Training of the Binary Relevance algorithm

```
Input: Training set $\mathcal{D}_{train}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Models $f_i(x)$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D}_i \leftarrow add\_label(X, \lambda_i)$
    $f_i(x) \leftarrow train(\mathcal{D}_i)$
end
```

This algorithm is fast, as it learns as many models as there are labels, but it does not use the interdependencies among the labels when predicting labels. It considers the multi-label problem as independent single-label problems.

Hence, this algorithm outperforms other multi-label algorithms only in two cases. (i) If there are only few dependencies among the labels, multi-label classifiers exploiting the dependencies do not perform better than BR. (ii) The data set consists of a large number of instances compared to a small number of labels. Thus, enough information from the features is given for each label, and no additional information from the labels is required to generate well-performing models.

The algorithm is fast and easy to apply, and it is the baseline classification method for multi-label problems. Many publications use this method when they do not recognize the problem as a multi-label classification task. Nevertheless, BR does not focus on one

3 Base classifiers in terms of single-label classifiers for the transformed data sets. Usually, more than one model is learned in each approach.

Algorithm 2.2: Prediction step of the Binary Relevance algorithm

```
Input: Test instance $x$, Threshold $\theta$
Output: Bi-partition $P_x, N_x$, Confidences $C_x$
$P_x$,$N_x$,$C_x \leftarrow \{\}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\hat{\lambda}_i \leftarrow f_i(x)$
    $C_x \leftarrow compose(C_x, \hat{\lambda}_i)$
    if $\hat{\lambda}_i > \theta$
    then
        $P_x \leftarrow add(P_x,\lambda_i)$
    else
        $N_x \leftarrow add(N_x, \lambda_i)$
    end
end
```

of the main properties of multi-label classification: labels are predicted in isolation from each other using only the features given in the data set. An algorithm addressing this shortcoming is 2BR.

2.1.4.2 2BR

2BR [93] tries to model the dependencies between labels by introducing a second level of prediction. On the base level, the data set is split in the same way as in the Binary Relevance method (see Algorithm 2.3), and classifiers are learned for each label. On the meta level, classifiers are learned for each label, but this time using the predicted labels as input instead of the real labels. When predicting a new instance, the base classifiers are used to predict the base level labels (see Algorithm 2.4). Subsequently, the multi-label classifier uses these predictions to predict the final label combination.

In the standard version of the algorithm, the complete training set is used to build the base level, and later the meta level. This can lead to meta level classifiers overfitting the training set. There are some approaches to avoid this problem. One way is to hold out a certain fraction of the training set and use it for the prediction of the meta level only. However, this can be problematic for small data sets. As an alternative, Tsoumakas *et al.* [93] suggested some form of pruning of the base level models before using them on the meta level.

The proposed method includes a preprocessing step before the generation of the models for the meta level (see Algorithm 2.5). Only labels with a strong enough correlation to the target label are included into the meta-level model for this target. The procedure is

Algorithm 2.3: Training of the 2BR algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Models $f_i(x)$ and $g_i(x), i \in \{1,...,q\}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D} \leftarrow add\_label(X, \lambda_i)$
    $f_i \leftarrow train(\mathcal{D})$
end
/* Perform cross validation to generate input data for second level
   of classifiers                                                   */
$\mathcal{D}' \leftarrow \{\}$
for $k \in 1,...,10$ do
    $\mathcal{D}_{train_k} \leftarrow train\_split(\mathcal{D}, k, 10)$
    $\mathcal{D}_{test_k} \leftarrow test\_split(\mathcal{D}, k, 10)$
    foreach $\lambda_i \in \mathcal{Y}_{train_k}$ do
        $D_{train_{k_i}} \leftarrow add\_label(\mathcal{X}_{train_k}, \lambda_i)$
        $f_i' \leftarrow train(\mathcal{D}_{train_{k_i}})$
        foreach $x_j \in \mathcal{D}_{test_k}$ do
            $\hat{\mathcal{Y}}' \leftarrow \{\}$
            foreach $\lambda_i \in \mathcal{Y}_{train_k i}$ do
                $\hat{\lambda}_i \leftarrow f_i'(x_j)$
                $\mathcal{Y}' \leftarrow compose(\mathcal{Y}', \hat{\lambda}_i)$
            end
            $\mathcal{D}' \leftarrow compose(\mathcal{D}', compose(x_j, \hat{\mathcal{Y}}'))$
        end
    end
end
/* Learn second level of classifiers                                */
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D}_i \leftarrow compose(\mathcal{X}, \hat{\mathcal{Y}})$
    $g_i \leftarrow train(\mathcal{D}_i)$
end
```

Algorithm 2.4: Prediction step of the 2BR algorithm

```
Input: Test instance $x$, Threshold $\theta$
Output: Bi-partition $P_x, N_x$, Confidences $C_x$
$P_x$,$N_x$,$C_x \leftarrow \{\}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\hat{\lambda}'_i \leftarrow f_i(x)$
    $\hat{Y}' \leftarrow compose(\hat{Y}', \hat{\lambda}'_i)$
end
foreach $\lambda_i \in \mathcal{Y}$ do
    $\hat{\lambda}_i \leftarrow g_i(\hat{Y}')$
    $C_x \leftarrow compose(C_x, \hat{\lambda}_i)$
    if $\hat{\lambda}_i > \theta$ then
        $P_x \leftarrow add(P_x, \lambda_i)$
    else
        $N_x \leftarrow add(N_x, \lambda_i)$
    end
end
```

based on the Pearson correlation coefficient between pairs of labels and a user-settable threshold.

The feature selection step of the second algorithm can improve the performance of 2BR. It adds an prepossessing step to the second level of prediction, in which unimportant pseudo labels are ignored by the classification. In addition to the gain in performance, the running time of the classifiers is improved, as there are fewer base models to be learned on the second level. Consequently, the dependencies between labels are taken into account, which improves the performance on data sets with strongly correlated labels.

2.1.4.3 Classifier Chains

Read *et al.* [77] introduced chains of classifiers. This algorithm seems to be one of the best-performing multi-label algorithms at the moment. The pseudocode is given in Algorithm 2.6 and Algorithm 2.7. Again, one single classifier is learned for each label. After each classifier is learned for a label λ_1, λ_1 is added to the feature set. The next classifier is learned on the newly generated feature set (consisting of the default feature set, and the first label λ_2). λ_2 is added to the feature set and the next classifier is trained. These steps are repeated until one model is learned for each label. For the prediction, the first classifier predicts the class. The prediction is used as additional input feature

Algorithm 2.5: Training of the Correlation Based Pruning of Stacked Binary Relevance Models algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$, Threshold $\theta$
Output: Models $f_i(x)$ and $g_i(x), i \in \{1, \ldots ,q\}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D} = X \cup \lambda_i$
    $f_i = train(\mathcal{D})$
end
/* Perform cross validation to generate input data for second level
   of classifiers                                                  */
$\mathcal{D}' \leftarrow \{\}$
for $k \in 1,...,10$ do
    $\mathcal{D}_{train_k} \leftarrow train\_split(\mathcal{D}, k, 10)$
    $\mathcal{D}_{test_k} \leftarrow test\_split(\mathcal{D}, k, 10)$
    foreach $\lambda_i \in \mathcal{Y}_{train_k}$ do
        $D_{train_{k_i}} \leftarrow add\_label(\mathcal{X}_{train_k}, \lambda_i)$
        $f_i' \leftarrow train(\mathcal{D}_{train_{k_i}})$
        foreach $x_j \in \mathcal{D}_{test_k}$ do
            $\hat{\mathcal{Y}}' \leftarrow \{\}$
            foreach $\lambda_i \in \mathcal{Y}_{train_k i}$ do
                $\hat{\lambda}_i \leftarrow f_i'(x_j)$
                $\mathcal{Y}' \leftarrow compose(\mathcal{Y}', \hat{\lambda}_i)$
            end
            $\mathcal{D}' \leftarrow compose(\mathcal{D}', compose(x_j, \hat{\mathcal{Y}}')$
        end
    end
end
/* Learn second level of classifiers                              */
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D}_i \leftarrow compose(\mathcal{X}, \hat{\mathcal{Y}})$
    $\mathcal{D}_i' \leftarrow select\_features(\mathcal{D}_i, \theta)$
    $g_i \leftarrow train(\mathcal{D}_i)$
end
```

for the next classifier and so on. Thus, there is a chain of classifiers, each member of the chain using the output of the previous classifiers for its prediction.

Algorithm 2.6: Training of the classifier chains algorithm

```
Input: Training set $\mathcal{D}$ with of features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Models $f_i(x)$
$\mathcal{D} \leftarrow \mathcal{X}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\mathcal{D} \leftarrow add\_label(\mathcal{D}, \lambda_i)$
    $set\_class(\mathcal{D}, \lambda_i)$
    $f_i \leftarrow train(\mathcal{D})$
end
```

The order of the labels in the chain affects its prediction and performance [77]. To obtain the best order of the labels and thus the best performing multi-label classifier, Read *et al.* [77] sampled 10 different orders uniformly at random and averaged their prediction to obtain the final prediction, resulting in an ensemble of classifier chains.

Algorithm 2.7: Prediction step of the classifier chains algorithm

```
Input: Test instance $x$, Threshold $\theta$
Output: Bi-partition $P_x, N_x$, Confidences $C_x$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\hat{\lambda}_i = f_i(x)$
    $x \leftarrow compose(x, \hat{\lambda}_i)$
    $C \leftarrow add(C, \hat{\lambda}_i)$
    if $\hat{\lambda}_i > \theta$ then
        $P \leftarrow add(P, \lambda_i)$
    else
        $N \leftarrow add(N, \lambda_i)$
    end
end
```

The method exploits label dependencies by including the predicted labels in the next model. At any point, it uses all the information available. Still, literature about the algorithm lacks some theoretical background and extensive analysis. It is not clear which label order is the best or why there are some orders which are better than others. Dembczyński *et al.* [16] gave classifier chains a first theoretical basis. Additionally, they extended classifier chains to probabilistic classifier chains which can be optimized to any loss function by estimating joint distributions. They interpret classifier chains as a greedy approximation of the most probable label set.

2.1.4.4 Ranking by Pairwise Comparison

Ranking by pairwise comparison by Hüllermeier *et al.* [47] learns a model for each pairwise comparison of labels. The pseudocode of the method is given in Algorithm 2.8 and Algorithm 2.9. Each label is pairwise compared with all other labels, and a new label is introduced for each comparison. If the label is positive in an instance, but the comparison label is negative, the new label is set to positive. If both labels are positive, the new label is negative. If the first label is negative, the instance is ignored for this label. On the new labels, models are learned independently. This results in $q(q-1)/2$ models, reducing the total number of labels. Using the predictions of these models, a ranking is produced by counting how often a label wins over another. A classification can be achieved by setting a threshold on the number of labels or probability gained from the single models.

This method exploits dependencies by using the pairwise comparison of labels. Nevertheless, more complex dependencies over multiple labels are not taken into account. Although the algorithm has a rather high complexity as $q(q-1)/2$ models need to be trained, the data sets for the single models are smaller than used for other multi-label classifiers. However, due to the complexity depending on the number of labels, it is not possible to apply the method to data sets with many labels.

2.1.4.5 Calibrated Label Ranking

Calibrated label ranking by Fürnkranz *et al.* [32] (see Algorithms 2.10, the prediction step is not given in pseudocode as it is basically identical to the one of ranking by pairwise comparison) is an extension to Ranking by pairwise comparison. The algorithm produces in addition to the comparison label a new virtual label, which is determined from a pairwise comparison of all labels to the new virtual label. Its value is simply the same value as of the respective other label. A final ranking of all labels is produced from the predictions of all the models, where labels ranking above the virtual label are set to positive, those below are set to negative.

The algorithm has the advantage that it introduces a kind of learned threshold which splits the predictions in positive and negative labels. This provides a more reliable way than to split by setting a threshold manually to split the labels. Nevertheless, the algorithm shares all other weaknesses with ranking by pairwise comparison. The complexity is rather high and it is not suitable for large data sets.

Algorithm 2.8: Training of the ranking by pairwise comparison algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Multi-label model $f(x)$
/* Create new set of labels                                        */
$\mathcal{Y}' \leftarrow \{\}$
for $i \in 1, \ldots q$ do
    for $j \in 1, \ldots q$ do
        if $i \neq j$ then
            /* introduce new label $\lambda_{ij}$ by using the combination of labels
               */
            $\lambda_{ij} \leftarrow 0$
            foreach $x \in \mathcal{D}$ do
                if $\lambda_i = 1$ then
                    if $\lambda_j = 1$ then
                        $\lambda'_{ij} \leftarrow 0$
                    end
                    if $\lambda_j = 0$ then
                        $\lambda'_{ij} \leftarrow 1$
                    end
                    if $\lambda_{ij} = 1 \wedge \lambda'_{ij} = 1$ then
                        $\lambda_{ij} \leftarrow 1$
                    else
                        $\lambda_{ij} \leftarrow 0$
                    end
                end
            end
            $\mathcal{Y}' \leftarrow add\_label(\mathcal{Y}', \lambda_{ij})$
        end
    end
end
$\mathcal{D}' \leftarrow compose(\mathcal{X}, \mathcal{Y}')$
/* Train a Binary Relevance model on the data                      */
$f(x) \leftarrow train_{BR}(\mathcal{D}')$
```

Algorithm 2.9: Prediction step of the ranking by pairwise comparison algorithm

```
Input: Test instance $x$
Output: Ranking $\mathcal{R}$
$\hat{Y}' \leftarrow f(x)$
$counts \leftarrow new\ Array[q]$
foreach $\lambda_{ij} \in \mathcal{Y}'$ do
    if $\lambda_{ij} = 1$ then
        $counts[i] \leftarrow counts[i] + 1$
    else
        $counts[j] \leftarrow counts[j] + 1$
    end
end
/* Sort labels according to counts                                */
$\mathcal{R} \leftarrow \{\}$
while $counts \neq []$ do
    /* Find index with highest count in $counts$                   */
    $i \leftarrow max\_index(counts)$
    $counts \leftarrow remove\_element(i, counts)$
    $\mathcal{R} \leftarrow add(\mathcal{R}, \lambda_i)$
end
```

Algorithm 2.10: Training of the calibrated label ranking algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Model $f(x)$, Ranking By Pairwise Comparison model $g(x)$
$\mathcal{Y}' \leftarrow \{\}$ for $i \in 1, \ldots, q$ do
    $\lambda'_i \leftarrow \lambda_i$
end
$\mathcal{D}' \leftarrow compose(X, \mathcal{Y}')$
$f(x) \leftarrow train(\mathcal{D}')$
/* learn Ranking by Pairwise Comparison model $g(x)$               */
$g(x) \leftarrow rpc\_train(\mathcal{D}_{train}))$
```

2.1.4.6 Label Power Set

Label power set creates a new multi-class problem from the original data set. The pseudocode is given in Algorithm 2.11 and Algorithm 2.12. Each label combination is used as a new class for the data set and a multi-class learner is applied. In the basic form of the algorithm, only combinations of labels which are present in the data can be predicted by the algorithm. This is problematic for data sets with a large number of unique label combinations.

Algorithm 2.11: Training of the label power set algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Model $f(x)$
$\mathcal{Y}' \leftarrow \{\}$
for $i \in 1, \ldots, m$ do
    /* introduce new class $\lambda'$ by using the combination of labels */
    $\lambda'_i \leftarrow \lambda_{i1,\ldots,\lambda_{iq}}$
    $\mathcal{Y}' \leftarrow add\_label(\mathcal{Y}', \lambda'_i)$
end
$\mathcal{D} \leftarrow compose(\mathcal{X}, \mathcal{Y}')$
$f(x) \leftarrow train(\mathcal{D})$
```

The complexity of the algorithm depends on the number of label combinations. If this number is large, so is the number of training examples for a single class, which makes the learning task for the multi-class algorithm more difficult. Another disadvantage of this algorithm is that it does not predict confidences for each label. Only the confidence for the complete label set is given. To overcome this downside, Read *et al.* introduced ensembles of pruned sets [77].

Algorithm 2.12: Prediction step of the label power set algorithm

```
Input: Test instance $x$
Output: Bi-partition $P_x, N_x$
$\hat{y}' \leftarrow f(x)$
foreach $\lambda_i \in \mathcal{Y}$ do
    if $\lambda_i \in \hat{\lambda}'_i$
    then
        $P_x \leftarrow add(P_x, \lambda_i)$
    else
        $N_x \leftarrow add(N_x, \lambda_i)$
    end
end
```

Ensembles of pruned sets by Read *et al.* [77] are based on the label power set approach. Ensembles of label power set models are trained on the training data. The prediction for each label is the combination of confidences for all predicted classes in which the label occurs. This gives a confidence for each label and thus results in a ranking for the labels. Furthermore, the label sets are pruned. If the frequency of a set is below a threshold, the set is split into smaller ones to occur at a minimum frequency.

2.1.4.7 Random k-Labelsets

Random k-labelsets (RAkEL) [98] was introduced to overcome some shortcomings of the label-power set method. RAkEL randomly creates subsets of the labels. The subsets are used as input for learning models independently and predictions on the labels are averaged. The main problem of label power set is the prediction of unknown label combinations. The model can only predict label combinations that are known in the training set. Using random sets of labels, new combinations are possible, as the existing label sets are split into multiple new sets. By recombining the new sets in the prediction, new combinations of labels become possible.

The learning process is straightforward (see Algorithm 2.13). Given the number of sets and their size, new label sets are generated from the training data. For each of the label sets, a new multi-label model is learned.

Algorithm 2.13: Training of the random k-labelsets algorithm

```
Input: Training set $\mathcal{D}_{train}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$, number of subsets $k$, size
       of subsets $k$
Output: Models $f_i(x)$
for $l \in 1, \dots, k$ do
    /* introduce new multi-label data set $\mathcal{X}'_l$ with a random label
       subset of size $k$                                                  */
    $\mathcal{D}_l \leftarrow compose(\mathcal{X}, random\_subset(\mathcal{Y}, k))$
    /* train multi-label model on new data set                             */
    $f_l(x) \leftarrow train(\mathcal{D}_l)$
end
```

The classification of a new instance (see Algorithm 2.14) averages the predictions of all models. If a label is only predicted by one model, only this prediction is used.

The algorithm was intended to help the label power set method. Nevertheless, any multi-label classifier can be used as a base algorithm. They can benefit from reducing the label space and using subgroups of labels. It is a similar approach to ensembles of classifier chains, as there too, random orders are used to improve classification. In the case of classifier chains, ensembles are used, since the best order of the chain is unknown. In the case of RAkEL, multiple random label combinations are used to exploit subsets with highly interdependent labels. Hence it is possible to improve the predictions of any multi-label classifier with this approach, not only when using label power set.

Algorithm 2.14: Prediction step of the random k-labelsets algorithm

```
Input: Test instance $x$
Output: Bi-partition $P_x, N_x$
for $l \in 1, \dots, k$ do
    $\hat{\mathcal{Y}}_l \leftarrow f_l(x)$
end
foreach $\lambda_i \in \mathcal{Y}$ do
    $\hat{\lambda}_i \leftarrow all\_predictions(\hat{\mathcal{Y}},i)$
    if $avg(\hat{\lambda}) > 0.5$ then
        $P_x \leftarrow add(P_x, \lambda_i)$
    else
        $N_x \leftarrow add(N_x, \lambda_i)$
    end
end
```

2.1.4.8 HOMER

RAkEL (see Section 2.1.4.7) uses multiple random combinations of labels. There are overlaps of labels in the subsets. This splits the problem into simpler sub-problems. Nevertheless, it is possible to perform more sophisticated splits of the data set. Dependencies and relations between the labels can be used to improve the performance. An algorithm approaching this problem is HOMER by Tsoumakas *et al.* [96].

HOMER generates a tree structure of the labels (see Figure 2.2). Its goal is to exploit hierarchical dependencies between the labels. Each node in the tree represents a set of labels. For each node, a multi-label model is learned, using all children of the node as new labels. In the final level, each leaf is one of the original labels of the data set. This algorithm can exploit dependencies between sets of labels and hence boost the total performance. Similar to RAkEL, it is a meta classifier for multi-label classification. It can use any multi-label classifier as base classifier.

When training HOMER, a tree structure has to be developed. This is done by a modification of the k-Means algorithm to generate balanced clusters [4]. The algorithm takes as input a set of labels and the data. It calculates k cluster centers, and then assigns a fixed number of labels to the centers. The closest values are assigned to the cluster, such that each cluster contains the same number of labels. The clustering step is done for each node in the tree with the set of labels of this node as input.

The prediction of the model is done level-wise. First, all latent labels (each summarizing one label sets) of the root node are predicted. Next, from all positive label sets, the child label sets are predicted. This is repeated until the algorithm reaches the leaves

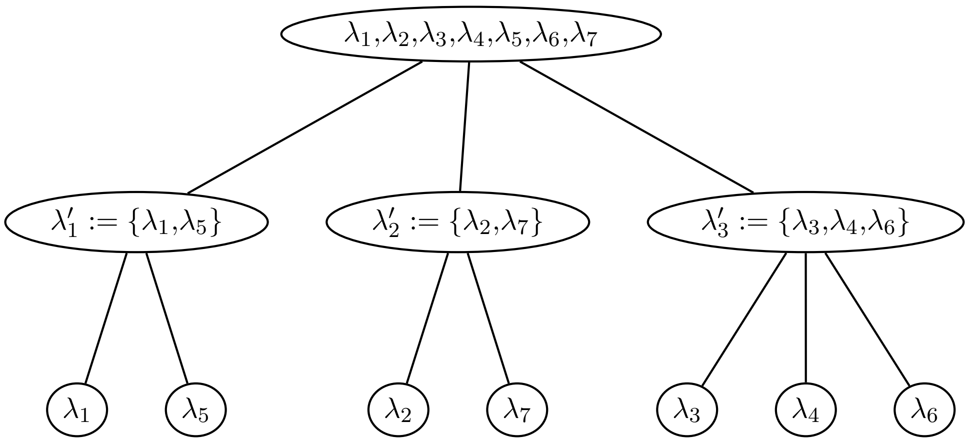

$\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6,\lambda_7$
$\lambda'_1 := \{\lambda_1,\lambda_5\}$
$\lambda'_2 := \{\lambda_2,\lambda_7\}$
$\lambda'_3 := \{\lambda_3,\lambda_4,\lambda_6\}$
λ_1
λ_5
λ_2
λ_7
λ_3
λ_4
λ_6

Figure 2.2: Example hierarchy tree of HOMER. Each node represents a set of labels. A model $f(x)$ is learned on each node, predicting all children of the node as new labels. In each node, only those instances are used, which contain at least one positive label from the labels corresponding to the node. λ_i is a label, and λ'_i is a meta-label, consisting of a label set.

of the tree. All positive leaves are returned as positive. So, by reducing the time and resources of the prediction step, this algorithm has to predict less labels. The learning step benefits from the tree structure in a similar way. For each node, only instances with at least one positive label in this label set are used for the training of this model and all following models. Additionally, the number of labels can be strongly reduced in all models. Hence, the size and training time of the models is smaller than when applying the multi-label models directly to the data set. Nevertheless, it depends on the structure of the tree, if there exist too many nodes, this advantage can be lost.

2.1.4.9 Further Transformation-Based Algorithms

Besides the previously presented methods, many other algorithms exist in the field of multi-label classification, that became highly popular in recent years. While the most important algorithms are presented above, there exist many more algorithms which cannot be covered here. Some methods try to reduce the label space and hence reduce the required resources, by exploiting dependencies between the labels. Tai and Lin [89] transformed the sparse label space and used the transformation in a method similar to binary relevance. Tenenboim-Chekina *et al.* [91] used clustering to identify dependent labels and learn models based only on them, using a mixture of binary relevance and the label power set method.

In a simpler approach, Boutell *et al.* [8] suggested two possible ways to handle multi-label data: (i) They randomly picked one label per instance and hence were able to consider the problem as single-label task. (ii) All instances with more than one label were ignored, again turning the problem into a single label task. Nevertheless, this approach completely ignores dependencies between the labels and brings only little advantage over the binary relevance method. Tsoumakas *et al.* [94] extended these approaches and suggested another transformation. Each label is used as a single class value, instances with multiple labels are copied and included multiple times in the data set, with one class label per positive label. This takes label dependencies into account as the algorithm learns the model based on the complete label information. Nevertheless, the model only predicts one single label and, depending on the chosen algorithm, the dependencies might not support the classification strongly.

2.1.5 Adaptation of Existing Algorithms

The second category of multi-label classifiers are classifiers based on existing single label algorithms. Numerous algorithms exist that are adapted to predict multiple labels at once. This procedure can be beneficial for classification, as dependencies among labels can be exploited not in the transformation step, but directly in the learning process of each model. Additionally, these models consume less resources in terms of memory and computing time. Transformation-based algorithms need to learn multiple base models, each being as complex as all models based on an adaptation of existing algorithms. On the other hand, these algorithms lack the flexibility of transformation-based algorithms. These classifiers can use an appropriate base model for the task. Different types of data sets require different approaches. For example, data sets that are not linearly separable benefit from non-linear SVMs, while linear models cannot generate meaningful models.

In contrast to the algorithms presented in Section 2.1.4, none of these algorithms use systems of classifiers. In general, only one model is built, which predicts multiple target values at once. Although such algorithms do not lie in the scope of this thesis, adaptation algorithms are commonly used in multi-label classification. Hence, in the following, we will give a short overview.

2.1.5.1 Multi-Label k Nearest Neighbor

Multi-Label k Nearest Neighbor [107] is a modification of the kNN algorithm. It is among the most commonly used multi-label algorithms. The main reasons for this are the low resource requirements and the low running time. The algorithm (see Algorithm

2.15 and Algorithm 2.16) first identifies the k nearest neighbors of a test instance x in the training set for which it uses the membership counting vector $\vec{C_x}$, that counts the neighbors of x with label λ positive, using $N(x)$ as the set of K nearest neighbors of instance x:

$$\vec{C_x} = \sum_{a \in N(x)} \vec{y_a}(\lambda) \tag{2.13}$$

$\vec{y}$ calculates the probability that a label is positive given the neighbors, this is calculated by using the maximum a posteriori principle. H_1^λ is the event that x has a positive label λ, H_0^λ the event that the label is negative. E_j^l with $j \in \{0, \dots, k\}$ is the event that within the k nearest neighbors of x there are exactly j instances with positive label λ.

$$\begin{align}
\vec{y_x} &= \operatorname*{argmax}_{b \in \{0,1\}} P(H_b^\lambda | E_{\vec{C_t}(\lambda)}^\lambda) \tag{2.14}\\
&= \operatorname*{argmax}_{b \in \{0,1\}} \frac{P(H_b^\lambda) P(E_{\vec{C_x}(\lambda)}^\lambda | H_b^\lambda)}{P(E_{\vec{C_x}(\lambda)}^\lambda)} \tag{2.15}\\
&= \operatorname*{argmax}_{b \in \{0,1\}} P(H_b^\lambda) P(E_{\vec{C_x}(\lambda)}^\lambda | H_b^\lambda) \tag{2.16}
\end{align}$$

To summarize, the algorithm predicts new labels similar to the single-label kNN algorithm. It works in three steps. (i) It finds the k nearest neighbors to a new instance. (ii) MLkNN calculates the membership counting vector for the instance from the neighbors. (iii) Using the maximum a posteriori principle, the result label set is predicted.

The algorithm has similar advantages and disadvantages as the single-label kNN. The training step is fast, at the expense of having longer computation times during prediction. The model can consume a lot of resources, as the whole data set has to be stored in it. On the other hand, if the dimensions of the data set are not too high (curse of dimensionality), the prediction can be rather quick. Additionally, the model is rather simple and provides a prediction that can be interpreted using the k nearest neighbors.

2.1.5.2 Further Algorithm Adaptation Methods

Besides the rather basic MLkNN, many further, more sophisticated algorithm adaptation methods exist. Nevertheless, they did not gain the same popularity as algorithms from the problem transformation category. The most intuitive approach might be the exten-

Algorithm 2.15: Training of the MLkNN algorithm

```
Input: Training set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$, number of neighbors $k$,
       smoothing parameter $s$
foreach $\lambda_i \in \mathcal{Y}$ do
    /* Calculate prior probabilities                                  */
    $P(H_1^\lambda) \leftarrow \frac{s+\sum_{i\leftarrow 1}^{m} \vec{y}_{x_i}(\lambda)}{s\cdot 2+m}$
    $P(H_0^\lambda) \leftarrow 1 - P(H_1^\lambda)$
    /* Calculate posterior probabilities                              */
    for $j \in 1,\ldots,k$ do
        $c[j] \leftarrow 0$
        $c'[j] \leftarrow 0$
    end
    for $i \in 1,\ldots,m$ do
        $\delta \leftarrow \vec{C}_{x_i}(\lambda_i) = \sum_{a\in N(x_i)} \vec{y}_a(\lambda_i)$
        if $\vec{y}_{x_i} \leftarrow 1$ then
            $c[\delta] \leftarrow c[\delta] + 1$
        else
            $c'[\delta] \leftarrow c'[\delta] + 1$
        end
        foreach $l \in 0,\ldots,k$ do
            $P(E_l^\lambda|H_1^\lambda) \leftarrow \frac{s+c[l]}{s\cdot(k+1)+\sum_{p\leftarrow 0}^{k} c[p]}$
            $P(E_l^\lambda|H_0^\lambda) \leftarrow \frac{s+c'[l]}{s\cdot(k+1)+\sum_{p\leftarrow 0}^{k} c'[p]}$
        end
    end
end
```

Algorithm 2.16: Prediction step of the MLkNN algorithm

```
Input: Test instance $x$
Output: Confidences $C$
$C = \{\}$
foreach $\lambda_i \in \mathcal{Y}$ do
    $\vec{C}_x(\lambda_i) \leftarrow \sum_{a\in N(x)} \vec{y}_x(\lambda)$
    $\vec{r}_x(\lambda_i) \leftarrow P(H_1^{\lambda_i}|E_{\vec{C}(\lambda_i)}^{\lambda_i}) = \frac{P(H_1^{\lambda_i})P(E_{\vec{C}_x(\lambda_i)}^{\lambda_i}|H_1^{\lambda_i})}{P(E_{\vec{C}_x(\lambda_i)}^{\lambda_i})} = \frac{P(H_1^{\lambda_i})P(E_{\vec{C}_x(\lambda_i)}|H_1^{\lambda_i})}{\sum_{b\in\{0,1\}} P(H_b^{\lambda_i})P(E_{\vec{C}_x(\lambda_i)}|H_b^{\lambda_i})}$
    $C \leftarrow add(C, \vec{r}_x(\lambda_i))$
end
```

sion of neural networks for the multi-label case. Zhang *et al.* [108] introduced neural networks using BP-MLL. Each input unit represents one feature of the data set, each output unit represents one label. The error function had to adapted to the multi-label case. Zhang *et al.* suggested to use a function approximating the ranking loss.

Another example of algorithm adaptation is AdaBoost.MH [83]. Strictly speaking, this algorithm is based on a data set transformation, and hence could be assigned to the transformation-based algorithm category as well. The algorithm transforms the data set to a binary classification task. Subsequently, Adaboost [82, 30] is used to learn the new single-label model. Adaboost is modified by optimizing the weights towards the Hamming loss function.

Support Vector Machines (SVMs) [14] are among the most popular and best performing classification methods in machine learning. Elisseeff *et al.* [26] adapted SVMs for multi-label classification. Their approach, called Rank-SVM, learns one classifier per label, but uses one margin for the whole data set. Similar to Adaboost.MH, this approach transforms a data set to a single-label form and then adapts the algorithm to the multi-label case by modifying central mechanisms of the algorithm: In the case of Adaboost, the optimization function, in case of SVMs, the margin calculation.

Another popular classification method is classification using decision trees. There are several possibilities to generate decision trees, one of the most widely used algorithms is the C4.5 algorithm by Quinlan *et al.* [74]. Multi-label C4.5 by Clare *at al.* [12] is an extension of this algorithm for multi-label problems. They simply adapted the entropy to be calculated over multiple labels and not only one label. The classification is done by allowing multiple labels in a leaf.

Predictive Clustering Trees [7] (PCTs) have been extended by Struyf *et al.* [88] to handle multi-label data. PCTs consider the trees as a hierarchy of clusters. Each node represents a cluster of the data set. The trees are induced using top-down induction. To handle multi-label data, a distance measures is introduced that allows calculating the distance over multiple labels.

2.2 Multi-Relational Learning

Multi-relational learning differs from traditional learning in the way the data are represented. In traditional learning, data are given in one relation [25]. In multi-relational data sets, the data are distributed over multiple relations. The relations are connected via connections on the attributes. There are several ways to use standard machine learning algorithms on this kind of data. The most common way might be to join the relations into one[4]. Using this, no information are lost. On the other hand, the data sets can become rather large, as parts of the data are duplicated.

A common solution to this problem is to use a special linkage between the relations. Relations can be connected in multiple ways. Instead of using all linked instances in other relations, only certain instances, or aggregated instances, are used. Many machine learning algorithms are based on distance calculation. Most common are the single, complete or average linkage. Single linkage simply uses the closest instance in the learning process, complete, the furthest, and average, the average distance between the instances. This becomes clear if one thinks of an algorithm such as k-nearest neighbor, where the k closest instances to a new one are used to classify a new instance by using the majority class of the neighbors. When using single linkage, the distance between two instances is calculated by using the instances themselves and the two linked instances from each main instance which are the closest to each other.

This thesis presents a special case of multi-relational learning: The data is stored in three relations: Two relations store one set of instances each. The third relation gives the class for the combinations of the instances from these relations. The schema is visualized in Figure 2.3. $\mathcal{X}'$ and $\mathcal{X}''$ represent the relations storing the instance sets, $\mathcal{Y}$ stores the target values of the mappings of the instances. This schema is rather common in real-world data [10, 31, 79]. Nevertheless, the most common solution for learning models is to join the three relations into one. This is done in a straightforward way. Each value y_{ij} of $\mathcal{Y}$ is merged with its corresponding rows x_i' from $\mathcal{X}'$ and x_i'' from $\mathcal{X}''$ into a new relation.

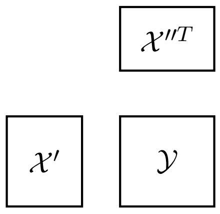


Figure 2.3: The special case of multi-relational learning that is covered in this thesis. $\mathcal{X}'$ and $\mathcal{X}''$ represent the mapping of two instance sets to their features. $\mathcal{Y}$ stores the class, a mapping of the instances in $\mathcal{X}'$ and $\mathcal{X}''$. $\mathcal{X}''$ is given transposed as $\mathcal{X}''^T$ to give a more natural visualization.

4 Note that this is only the case for certain data, i.e., data given in relations with a n:1 relationship.

This type of problem is also closely related to collaborative filtering. Algorithms for collaborative filtering are used in recommender systems [1], which are software systems that recommend items, e.g., books, music or movies, to a person, that are interesting to this person. Collaborative filtering approaches recommend items based on his or her previous interest in other items[5].

Piccart *et al.* [72] proposed an algorithm closely related to ClassFact, that is presented in Section 3.2.1. They define the problem of collaborative filtering as two-way learning[6]. Piccart *et al.* proposed to learn models for each relation, $\mathcal{X}'$ and $\mathcal{X}''$. The predictions of both learners can be combined, similarly to ensemble predictions. The data sets used in this publication take one relation, e.g., a mapping between CPUs and their performance in benchmark programs, and use a part of the data as input, i.e. they use the performance of a subset of the CPUs in a subset of the benchmarks and predict the performance of new CPUs in new benchmarks from the known benchmarks.

2.3 Biodegradation Pathway Prediction

The area of biodegradation pathway prediction aims to predict the environmental fate of chemicals that are in some form released to the environment. This includes the prediction of the biodegradation products as well as complete biodegradation pathways of chemical compounds. The main goal is to prevent the disposal of chemicals that degrade into potentially hazardous (i.e., toxic) degradation products.

At the moment, the most common approaches predicting the environmental fate of chemicals are based on expert knowledge. Commonly, products and pathways are generated manually. To generate complex pathways, knowledge-based approaches, such as the University of Minnesota Pathway Prediction System (UM-PPS) for microbial biodegradation [43, 27] or METEOR for the prediction of mammalian metabolism [36], are available. The major problem with these knowledge-based systems is the combinatorial explosion. In general, these systems predict every possible degradation product, even if the product is unlikely to occur in reality. Hence, the number of pathways grows exponentially with a high branching factor. Therefore, methods limiting the results of the prediction are required.

Approaches based on machine learning to predict the environmental fate of chemicals

5 Another category in recommender systems are content-based algorithms, which consider features of the items for the prediction of interesting items. However, these systems are beyond the scope of this thesis.

6 Two-way, as features are given from two dimensions or relations.

are rather uncommon. Hence, there is no approach to incorporate large classifier systems in the process of predicting biodegradation pathways. Nevertheless, the task provides the structure to use systems of classifiers. In each degradation step, multiple potential biodegradation products occur. Each product occurs with a certain probability, depending on the structure and the conditions of the degradation. Machine learning algorithms using multiple classifiers to provide probabilities for multiple target values can prune unlikely pathways and prevent the combinatorial explosion occurring in expert-based systems.

2.4 Predictive Toxicology

Predictive Toxicology is the task of predicting toxic effects of chemicals [41]. The goal is to develop *in silico* models for chemical compounds which predict toxic endpoints. The endpoints are toxic effects, e.g., kidney failure or brain tumor. Hence, the models aim to predict the outcome of *in vivo* experiments with animals. This becomes more and more needed as regulations such as REACH [75] in the European Union require chemicals to be tested for toxic effects to be released. This would require many animal experiments to be carried out, which is ethically disputed. Another drawback of *in vivo* experiments are the high financial costs.

In silico models are usually trained on data sets containing known toxic effects from *in vivo* experiments. As these experiments are not easy to perform, the data sets tend to be rather small in terms of the number of chemicals. For example, the ToxCast™ data set (see Section 5.1) consists of only 309 instances. This is not an exception, but rather typical for toxicological data sets. Nevertheless, the ToxCast™ data set is rather large in terms of features and target values. As the target values clearly depend on each other (toxic effects for related species are included, different toxic effects on the same organs are in the data set), the data set tends to be suitable for sophisticated algorithms exploiting these dependencies, multi-label algorithms.

Despite the advantages of using multi-label classifiers in predictive toxicology, they have hardly been used in this domain yet. At the moment, there exists no publication using multi-label approaches in predictive toxicology. The only reference to multi-label classification is the use of predictive clustering trees [7] on a preliminary analysis of the ToxCast™ data set. This work was presented at the ToxCast™ Data Analysis Summit in May 2009, but was never officially published. The standard approach in this area is a propositional classification of single endpoints. The main focus lies on the calculation of descriptors of the chemicals to improve the performance.

CHAPTER 3

Large Classifier Systems

In this chapter, I will present approaches using large classifier systems. Traditional supervised machine learning approaches focus on learning one classifier for one data set, predicting one target value. This target value is predicted from a set of features, which describe the instances. Nevertheless, data can be described in a more complex way. Features can be distributed over multiple relations, several target values might be possible for one instance. A large classifier system uses multiple classifiers to predict one or more target variables at once. Depending on the problem definition, there exist several ways to combine multiple classifiers. In the context of this thesis, I cannot present all conceivable types of large classifier systems, hence, I will focus on multi-label classification and multi-relational classification.

A popular example of a large classifier system is multi-label classification[7]. Multi-label problems have multiple interdependent target variables. The challenge is to predict the variables together, exploiting their dependencies. In other words, the model needs to use information in the prediction that needs to be predicted itself. This kind of data is becoming extremely popular, common examples are tags for objects like texts, products in online stores or images. These objects need to be labeled automatically in many scenarios. This applies for example to texts on web pages, which need to be labeled automatically with descriptions of the content based on the text. Online stores want to assign their products to the right category.

Another application of large classifier systems includes certain multi-relational problems. In multi-relational data mining, the data set is distributed over multiple relations. A simple example is given in Section 3.2.1. Buchwald *et al.* [10] generated a data set

7 Related work on this topic is described in Section 2.1

describing the interaction between kinases and inhibitors. An inhibitor can bind to a kinase and prevent it from functioning. Thus, the data can be stored in three relations, one describing the kinases, one describing the inhibitors, and one storing if the kinases bind to the inhibitors[8].

Multi-label classification and multi-relational classification are two popular examples of large classifier systems. Nevertheless, this area is not limited to these. Although not covered in this thesis, other important examples are multi-class classification [40, 90, 2] and, as a specialization, ordinal classification [29, 60, 23]. Multi-class classification is similar to multi-label classification as one instance has several possible target values. Nevertheless, the target values are mutually exclusive, only one target value is assigned per instance. Simple approaches to this classification task are combining multiple binary classifiers to predict the correct class. A specialization of this task is ordinal classification. In this case, an instance can be assigned to one of several classes, same as in multi-class classification, with the only difference being that the classes have an order, that can be used to improve the classification.

The remainder of this chapter is organized as follows. I first give an introduction to multi-label classification, followed by a description of a new algorithm for multi-label classification that uses Boolean matrix decomposition to exploit the label dependencies. Subsequently, I present extensive experimental results to evaluate the algorithm. In the next section, I propose a new method to approach certain multi-relational problems using Boolean matrix decomposition, which is based on the previously presented multi-label algorithm. Finally, the experimental results from the evaluation on three data sets are presented.

3.1 Multi-Label Classification

Multi-label classification is a supervised learning task where each instance is assigned a subset of a set of possible labels. Multi-label classification algorithms try to exploit the relationships or correlations among labels to improve overall classification accuracy. It is the most natural classification setting for a number of application areas including text mining [83, 34, 32], image and video classification [8, 20], and bioinformatics [26, 103]. The notation used in this section is visualized in Figure 3.1.

One possible approach to address multi-label classification is to decompose the output

8 In the approach by Buchwald *et al.* [10], all relations were merged into one single relation. This is commonly done and leads to duplicate entries in the relation and higher computation times.

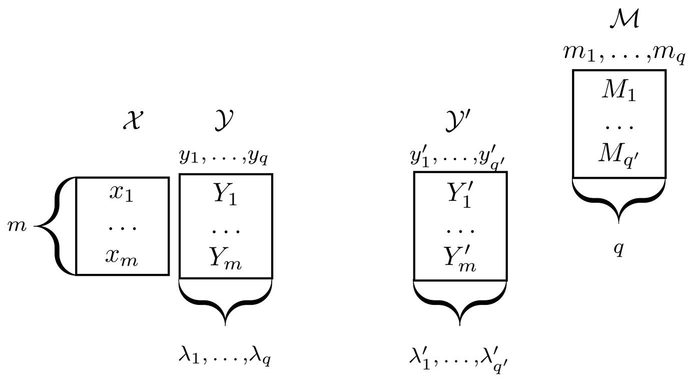


Figure 3.1: Notation used in this section. $\mathcal{Y} = \{\lambda_1, \dots, \lambda_q\}$ is the set of output labels. $\mathcal{S} = \{(x_1, Y_1), (x_2, Y_2) \dots (x_m, Y_m)\}$ with the feature vector $x_i \in \mathcal{X}$ and the label set $Y_i \subseteq \mathcal{Y}$ gives the instances of a multi-label data set. For the matrix factorization, the labels $\lambda_1 \dots \lambda_q$ are ecoded in a Boolean matrix. Each instance represents one row in the matrix $\mathcal{Y}$, a value $y_{ij}, i \in 1, \dots m, j \in 1, \dots q$ is set to 1, if label λ_j is positive for instance x_i. If λ_j is negative for instance x_i, the value is set to 0. The decomposition matrix $\mathcal{Y}'$ consists of the vectors y'_j with $j \in \{1 \dots q'\}$. The bi-partition of the labels is given by the set of predicted labels P_x and not predicted labels N_x.

matrix into its main components in the form of latent or pseudo labels that capture the most important information of the original labels along with their dependencies. Given such a decomposition of the original output matrix, it is easy to obtain predictions for the original labels by matrix multiplication. The only approach along those lines so far uses singular value decomposition (SVD) for the factorization of the output matrix [89]. In this section, we propose to use *Boolean matrix decomposition (BMD)* [68] for this purpose instead, which accounts for the Boolean data type of the class labels as well as the associations among them. As it turns out in experiments, the approach based on BMD performs favorably compared to the SVD-based approach across a wide range of data sets and performance measures.[9]

The remainder of this section is organized as follows. First I give a short introduction to the algorithms used as a basis for the new approach. Building on that, the new method, called MLC-BMaD (Multi-Label Classification using Boolean Matrix Decomposition), is described in detail. Next, performance measures are introduced. Finally, the new method is compared to and evaluated against benchmark algorithms across several standard multi-label data sets.

9 In preliminary experiments, we also tested principal component analysis (PCA) and independent component analysis (ICA) for the factorization of the output matrix, but ultimately the approach based on Boolean matrix decomposition is conceptually most appealing as well as best-performing in practice.

3.1.1 Related Work

The binary relevance method, one of the simplest multi-label schemes, divides the problem into multiple single-label classifications. The data set of q labels is split into as many data sets. Each single data set is assigned one label λ_i as class, which results in one single label problem per label. Classifiers are learned for each data set. To classify a new instance, each single label classifier is applied to the instance, and the resulting label set is the combination of all single classifications.

The algorithm is fast and easy to apply. However, it does not focus on one of the main properties of multi-label classification: Dependencies among labels are not taken into account, labels are predicted in isolation from each other using only the features given in the data set. An algorithm addressing this shortcoming is BR2.

BR2 aims to model the dependencies between labels by introducing a second level of predictions. On the base level, the data is split in the same way as in the Binary Relevance method. Classifiers are learned for each label. On the meta level, again classifiers are learned for each label, but using only the predicted labels as input. When predicting a new instance, the base classifiers are used to predict the base level labels. Using the prediction of these classifiers, the meta level classifiers are used to predict the final label combination.

The advantage of this procedure is that the dependencies among labels are taken into account. This improves the performance on data sets with strongly correlated labels. In the standard version of the algorithm, the complete training set is used to build the base level, and later the meta level. This can lead to meta level classifiers overfitting the training set. There are some approaches to avoid this problem. One way is to hold out a certain fraction of the training set and use it for the prediction of the meta level only. However, this can be problematic for small data sets. As an alternative, Tsoumakas *et al.* [93] suggested some form of pruning of the base level models before using them in the meta level.

The method proposed by Tsoumakas *et al.* [93] includes a preprocessing step before the the models for the meta level are generated. Only labels with a strong enough correlation to the target label are included into the meta-level model for this target. The procedure is based on the Pearson correlation coefficient between pairs of labels, and a user-settable threshold.

Recently, three methods related MLC-BMaD were published. Tai and Lin [89] transform the typically sparse label space and used the result in a method similar to Binary Relevance. This work was based on Hsu *et al.* [45], who used compressed sensing to exploit the sparsity of the label space. Tenenboim-Chekina *et al.* [91] use clustering to identify dependent labels and learn models based on them, using a combination of binary relevance and the label power set approach.

Another method, using non-negative matrix factorization, was introduced by Liu *et al.* [61]. However, this approach differs widely from the approach proposed here. Not the labels themselves are factorized, but the assignment of labels to a new instance is considered as a optimization problem. This problem is framed as non-negative matrix factorization under a linear constraint. The similarity matrix between predicted labels and labels in the training set is factorized into three matrices, one of them being the labels in the training set, one being the labels in the instances to predict, and the third matrix representing the class similarities. The method minimizes the difference between this similarity matrix and the similarity matrix between the features. Thus, this method uses a different perspective on multi-label classification and is in fact not closely related to the approach presented here.

The performance of the above methods depends strongly on characteristics of the data sets. In our experiments, most algorithms performed best on smaller data sets with only few labels and many instances. Typical multi-label problems, however, have a large number of labels and rather few instances in general. Algorithms performing well on these are more complex and tend to consume a lot of memory and running time. The contribution of this section is a simple algorithm that still performs well on complex data sets with strong dependencies among the labels.

3.1.2 Multi-Label Classification using Boolean Matrix Decomposition

The proposed method can be viewed as adding a preprocessing and a postprocessing step to the BR algorithm. Using Boolean matrix decomposition according to Miettinen *et al.* [68], the label matrix is factorized into a factor matrix of latent labels and a factor matrix of the interdependencies among these labels. Instead of learning models for the actual labels, models are learned for the latent labels. The final labels are predicted by Boolean matrix multiplication using the second factor matrix, that contains the interdependencies among the labels. The advantage of this method is the introduction of a new level of abstraction, which represents the data in a more compact way than the original label space.

In the following, I will go into the details of the proposed method. In the following subsection, I will recall the underlying Boolean matrix decompostion method. In the two subsequent subsections, the training and test phases will be explained.

3.1.2.1 Boolean Matrix Decomposition

Generally, the goal of a matrix decomposition is to represent a given matrix as the product of two or more factor matrices. In our case, we are given the $m \times q$ matrix of labels $\mathcal{Y}$, which we wish to factorize into the $m \times q'$ matrix of latent labels $\mathcal{Y}'$ and the second $q' \times q$ factor matrix $\mathcal{M}$ needed to reconstruct the original labels (see Table 3.1). When applying matrix decomposition to labels, the matrix to decompose is Boolean. Thus, all factor matrices need to be Boolean as well. The multiplication of the factor matrices is done in the Boolean domain (0s and 1s being interpreted as truth values), therefore $1 + 1 = 1$.

MLC-BMaD uses the Boolean matrix decomposition approach proposed by Miettinen *et al.* [68], which first calculates associations between all pais of columns of the matrix. In our case, associations are calculated between label columns. In the second step, candidate basis vectors are selected from the associations. Finally, a set of basis vectors is selected from the basis in a greedy manner.

For the first step, an internal association matrix A is created, that contains the associations between the columns. Each row and column in A represents a column in the base matrix $\mathcal{Y}$. The binary values indicate if there is an association between two columns. There is an association between the columns or labels if $c(i \Rightarrow j) \geq \tau$, where confidence $c(i \Rightarrow j) = \langle y_i, y_j \rangle / \langle y_i, y_i \rangle$, with $\langle \cdot, \cdot \rangle$ denoting the vector inner product operation.

Each row of the resulting matrix A is a candidate for being a basis vector. τ is a parameter of the algorithm and controls which attributes are included in the basis vector candidates. The basis vectors are selected in a greedy way. Initially, $\mathcal{M} = 0^{q' \times q}$ and $\mathcal{Y}' = 0^{m \times q'}$. A row is selected from matrix A and inserted as a row into matrix $\mathcal{M}$. At the same time, a column y_i' is generated maximizing the function:

Table 3.1: Example Boolean matrix decomposition of a 5×5 matrix $\mathcal{Y}$ into matrix $\mathcal{Y}'$ and M. The decomposition produces matrices so that $\mathcal{Y} \approx \mathcal{Y}' \cdot \mathcal{M}$. The reconstruction errors are indicated by gray boxes.

$$\mathcal{Y} = \left.\overbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & \boxed{0} & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & \boxed{0} & 1 & 0 & 1 \end{pmatrix}}^{q}\right\} m \Rightarrow \mathcal{Y}' = \left.\overbrace{\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}}^{q'}\right\} m, \quad \mathcal{M} = \left.\overbrace{\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}}^{q}\right\} q'$$

$$cover(\mathcal{Y}', \mathcal{M}, \mathcal{Y}) = |\{(i,j) : y_{ij} = 1, (\mathcal{Y}' \cdot \mathcal{M})_{ij} = 1\}| - |\{(i,j) : y_{ij} = 0, (\mathcal{Y}' \cdot \mathcal{M})_{ij} = 1\}| \tag{3.1}$$

The function rewards cases of 1s in agreement and penalizes cases of disagreement, when 1s are incorrectly obtained. The combination of a row and column maximizing this function is inserted in the decomposition matrices $\mathcal{Y}'$ and $\mathcal{M}$. The original paper introduced weights for the two different errors used in the function above. In our experiments, we set both weights to one. The time complexity of constructing the association matrix is $O(mq^2)$. A single basis vector can be calculated in $O(mq^2)$. Therefore, the overall complexity is $O(q'mq^2)$. The algorithm is described in detail in Algorithm 3.1.

Algorithm 3.1: *Boolean_decomposition*

```
Input: Boolean matrix $\mathcal{Y}$, threshold $\tau$
Output: Boolean matrices $\mathcal{Y}'$ and $\mathcal{M}$
$A \leftarrow 0^{q \times q}$
$\mathcal{Y}' \leftarrow 0^{m \times q'}$
$\mathcal{M} \leftarrow 0^{q' \times q}$
for $i \leftarrow 1, \dots, q$ do
    for $j \leftarrow 1, \dots, q$ do
        if $c(i \Rightarrow j) \geq \tau$ then
            $a_{ij} \leftarrow 1$
        end
    end
end
for $l \leftarrow 1, \dots, q'$ do
    row $l$ of $\mathcal{M} \leftarrow$ row $i$ of $A$
    $y'_j \leftarrow \{0,1\}^{q'}$ maximizing $cover(\mathcal{Y}', \mathcal{M}, \mathcal{Y})$
end
```

The quality of the decomposition can be calculated using the relative reconstruction error:

$$relReconError = \frac{|\{(i,j) : y_{ij} = 1, (\mathcal{Y}' \cdot \mathcal{M})_{ij} = 0\}| + |\{(i,j) : y_{ij} = 0, (\mathcal{Y}' \cdot \mathcal{M})_{ij} = 1\}|}{m \cdot q} \tag{3.2}$$

The relative reconstruction error calculates the ratio of incorrectly occurring 1, or 0, in the matrix generated by multiplying the two matrices produced by the decomposition step.

3.1.2.2 Learning the Model

The algorithm for training the model (see Algorithm 3.2) first extracts the labels from the data set. The resulting binary $m \times q$ matrix is decomposed into a $q' \times q$ matrix $\mathcal{M}$ and a $m \times q'$ matrix $\mathcal{Y}'$. Matrix $\mathcal{Y}'$, having the same number of rows m as there are instances in the data, is used in the next step. For the base level, the matrix is combined with the features of the data set. This data is used as input in a Binary Relevance multi-label algorithm. Matrix $\mathcal{M}$ is stored as part of the model. It is used to get the final predicted labels by multiplying the predicted latent variables with it.

The upper matrix $\mathcal{M}$ contains the associations between labels. A row is the association vector, containing all associations between one label and all others. The associations contain the information of the dependencies between the labels. The left matrix $\mathcal{Y}'$ contains one row per instance. The columns are a representation of a subset of the labels. They are approximately the same as the selected labels. In $\mathcal{M}$, every entry tells if there is an association between this label and the other label. As a label has an association value of 1 to itself, there is always a 1 in the row representing that label. When multiplying the matrices, the value in each cell is true if there is at least a 1 in a column of $\mathcal{Y}'$ and a 1 in the corresponding row and column of $\mathcal{M}$. The calculation of each value $\hat{y}_{ij}$ in $\hat{\mathcal{Y}}$ can be seen as Boolean expression:

$$\hat{y}_{ij} = y'_{i1} \wedge m_{1j} \vee y'_{i2} \wedge m_{2j} \vee \ldots \vee y'_{iq'} \wedge m_{q'j} \tag{3.3}$$

An entry in $\mathcal{M}$ is only 1 if either the row represents the same label as the column of $\mathcal{M}$ or the label indicated by the column of $\mathcal{M}$ frequently occurs together with the label represented by the row of $\mathcal{M}$ (which is also represented by the corresponding column of $\mathcal{Y}'$). Thus, we build classifiers for a subset of the labels and fill the remaining labels by using the dependencies between them.

The time complexity depends quadratically on the number of labels (see subsection 3.1.2.1), which lies between Binary Relevance (linear) and label power set (exponential). The approach optimizes the confidence of the multiplication of the decomposed matrices in the decomposition step. Therefore, MLC-BMaD effectively optimizes the accuracy [17].

Algorithm 3.2: Pseudocode for training the MLC-BMaD classifier

```
Input: Data Set with features $\mathcal{X}$ and labels $\mathcal{Y}$
Output: Boolean matrix $\mathcal{M}$ and set of models $f$
$(\mathcal{Y}', \mathcal{M}) \leftarrow Boolean\_decomposition(\mathcal{Y})$
foreach $y'_i$ do
    /* train model $f_i$ for $y'_i$                                   */
    /* $compose(\mathcal{X}, y'_i)$ creates a data set with the features and label $y'_i$
       as target value                                                 */
    $\mathcal{D}_i \leftarrow compose(\mathcal{X}, y'_i)$
    /* $train(\mathcal{D}_i)$ trains a single label model from data set $\mathcal{D}_i$   */
    $f_i \leftarrow train(\mathcal{D}_i)$
end
```

Algorithm 3.3: Pseudocode for predicting labels of a new instance using the MLC-BMaD classifier

```
Input: Instance with features $x$
Output: Bipartition $P_x, N_x$
for $i \leftarrow 1, \ldots, q'$ do
    $\hat{y'_i} \leftarrow f_i(x)$
end
$P_x, N_x \leftarrow \{\}$
for $i \leftarrow 1, \ldots, q$ do
    $\hat{y}_i \leftarrow false$
    for $j \leftarrow 1, \ldots, q'$ do
        $\hat{y}_i \leftarrow \hat{y}_i \vee ((\hat{y}'_j = 1) \wedge m_{ij})$
    end
    if $\hat{y}_i$ then
        $P_x \leftarrow add(P_x, \lambda_i)$
    else
        $N_x \leftarrow add(N_x, \lambda_i)$
    end
end
```

3.1.2.3 Applying the Model to New Instances

When predicting a new instance, first the base level classifiers are used to predict the latent labels. Each such latent label is predicted independently, using an approach similar to the Binary Relevance method. The predicted latent labels are then combined into a vector. The vector is multiplied with the stored matrix $\mathcal{M}$. The pseudocode in Algorithm 3.3. shows how Boolean multiplication is applied to come up with a predicted label set.

3.1.3 Evaluation

We evaluated the method on several publicly available multi-label data sets. We performed a *hold-out* validation on each data set and examined the performance. We used *hold-out* sampling due to the complexity of multi-label classification which causes a big overhead in running times. This makes the use of *cross-validation* impractical. The performance is compared to six multi-label classifiers implemented in the Mulan[10] [97] library. The classifier on the base level was a support vector machine [14]. The parameters for the learner were set to the default values in the WEKA[11] workbench [37] (complexity constant $c = 1$ and linear kernel). The class probability estimates of the classifier are thresholded against 0.5 before matrix multiplication.

The Boolean matrix decomposition introduces two parameters: The size of the decomposition vector, i.e., the number of columns in $\mathcal{Y}'$, and the threshold for the association between two labels. If the calculated confidence between two labels is smaller than the threshold, there is no association, otherwise, there is one. To select the best parameters, we conducted an additional *inner hold-out* sampling and tested the performance of every parameter setting. The size of the decomposition matrix was set to the minimum of 2, as a decomposition into only one column cannot reconstruct the original matrix. The maximum is set to either 32 due to that limitation of the used library, or, if smaller, to the number of labels. The step size for this parameter is 1. The threshold for the association matrix was varied between 0.1 and 1.0 in steps of 0.1.

Note that k-fold *cross-validation* or *hold-out* validation is not the best way to validate a multi-label algorithm. Sampling cannot be stratified, as the distribution of each label is too different from the distributions of the other labels. Problems also come with data sets with missing values in the labels as well. The only fair way to validate multi-label problems would be *leave-one-out cross-validation*, but this procedure is not practical in most cases. The complexity of this is far too high, especially for problem transformation algorithms. At least one classifier would need to be learned per label, and this would have to be repeated for every single example held out during *leave-one-out cross-validation.*

10 `http://mulan.sourceforge.net`
11 http://www.cs.waikato.ac.nz/ml/weka/

3.1.3.1 Implementation

We implemented the algorithm in the Mulan library, which is a multi-label library based on the WEKA workbench. It provides most required methods to handle and access multi-label data. For one data set (*Llog*) we adapted Mulan to handle missing values in the labels.

3.1.3.2 Data Sets

For the evaluation process we use six publicly available standard benchmark multi-label data sets. Statistics on the data sets are given in Table 3.2. To describe the data sets, we provide three typical multi-label statistics. The first one is cardinality, which is the average number of labels per instance. This is a good measure of the dependencies between the labels. A cardinality close to one shows there are almost no dependencies in the labels which are represented in the data set, as there is only one label on average present in an instance. The label density is label cardinality divided by the total number of possible labels. This can be considered as the ratio of labels per instance. Finally, we give the number of distinct label combinations in a data set. This can be used as an important measure when choosing the classification algorithm. Methods using label combinations as new classes can degrade in performance when the number of distinct label combinations is too high. Classifiers like the label power set method struggle to learn meaningful models for such data.

The *Emotions* data set consists of songs and emotions assigned to them as labels. Tsoumakas *et al.* [101] extracted several audio features from a 30 second sample of each song. Emotion labeling was done using the Tellegen-Watson-Clark model. The *Scene*

Table 3.2: Summary statistics of used multi-label data sets. Cardinality is the average number of positive labels per instance. Density is the cardinality averaged over the number of labels. This gives the ratio of positive labels per instance. Distinct is the number of distinct label combinations in the data set.

Name	Domain	Instances	Attributes	Labels	Cardinality	Density	Distinct
Emotions	music	593	72	6	1.869	0.311	27
Scene	image	2407	294	6	1.074	0.179	15
Yeast	biology	2417	103	14	4.237	0.303	198
Enron	text	1702	1001	53	3.378	0.064	753
Llog	text	1400	1003	75	1.180	0.016	304
Medical	text	978	1449	45	1.245	0.028	94

data set was first used by Boutell *et al.* [8]. It is a collection of images and assigned categories like *beach*, *sunset* or *mountain*. The features are extracted from the image data. The labels of the *Yeast* data set are the functional classes according to Funcat[12]. The features are generated by micro array expression data.

Enron is extracted from the Enron Email Analysis Project[13]. The messages were labeled with hierarchical categories for the emails. Top level categories are *coarse genre, included/forwarded information, primary topics* and *emotional tone*. The data set was already used in multi-label analysis by Read *et al.* [78]. *Medical* by Pestian *et al.* [71] is a collection of medical documents. The labels are insurance codes, the features are extracted from a free text overview of the patients symptom history and prognosis. *Llog* was extracted by Jesse Read [76] from the Language Log Forum[14]. The Language Log Forum is hosted by the University of Pennsylvania and is a collection of discussions on language. The labels are 75 topics, e.g. *language_and_politics, errors, humor, computational_linguistics*.

The first three data sets (*Emotions*, *Scene* and *Yeast*) have many instances mapped to only few labels. These are rather small data sets. The other three data sets (*Enron*, *Medical* and *Llog*) have few instances mapped to rather many labels, so that there are possibly more dependencies between the labels; therefore it makes more sense to take dependencies into account. The algorithms cannot learn many dependencies from only 6 or 14 labels, but around 50 labels may offer a greater chance to learn from them.

3.1.3.3 Algorithms

We first compared MLC-BMaD to six multi-label algorithms as implemented in the Mulan library. For each algorithm, we used the base classifiers that seemed to achieve the best performance over all data sets. Multi-label kNN [107] is a modification of the kNN algorithm. For any new instance, the k nearest neighbors in the training set are identified. Based on statistical information on the labels sets of the neighbors, the maximum a posteriori principle is used to set the labels for the new instance. Label power set uses each combination of labels present in the data set as a single label. With this, a multi-class model is used to predict the label combination.

Binary relevance is described above in Section 2.1.4.1. As a base classifier, we used the J48 classifier. Multi-Label Stacking by Tsoumakas *et al.* [93], which includes correlation-

12 `http://mips.helmholtz-muenchen.de/proj/funcatDB/search_main_frame.html`
13 `http://bailando.sims.berkeley.edu/enron_email.html`
14 `http://languagelog.ldc.upenn.edu/nll`

based preprocessing, is explained in detail above in Subsection 2.1.4.2. The base classifier is kNN, the meta classifier is logistic regression.

The include labels algorithm transforms each label into two new attributes: one nominal attribute giving the label name and one binary attribute encoding the presence of this label. Thus, each instance is transformed into q new instances, one for each label. Then a single label classifier is learned for the binary attribute indicating the presence or absence of the label. To predict a new instance, it also needs to be transformed into q new instances, one for each label. The base classifier here is J48.

Calibrated label ranking by Fürnkranz *et al.* [32] produces a new virtual label. A pairwise comparison of the labels is conducted, and models are learned for these comparisons. The virtual labels are always simply the negation of the respective other label. A final ranking of all labels is produced from the predictions of all the models. Labels ranking above the virtual label are set to positive, and those ranking below are set to negative. Again, the base classifier is J48.

Additionally, we compared our approach to the closest approach in the literature, principle label space transformation (PLST), by Tai and Lin [89], which is based on singular value decomposition (SVD). The size of the transformed label space was set to 100%, as this seems to give the best performance for this method, according to the authors. For this algorithm, we used the implementation of Tai and Lin [89] and wrapped it into Mulan.

3.1.4 Results

Overall, MLC-BMaD performs well compared to other algorithms (see Table 3.3). There is no single classifier which is always better than MLC-BMaD. The results strongly indicate that the classifier performs better on data sets with many labels, high cardinality and fewer instances, which is illustrated in Figure 3.2. The data set with the fewest labels is on the left, the data set with most labels is on the right. The number of labels is increasing from left to right. At the same time, the number of times MLC-BMaD performs significantly better than the other classifiers increases, while the number of times other classifiers perform significantly better decreases. This observation complies with the fact that the Boolean matrix decomposition can easily produce a good decomposition of the labels when being applied to data sets with more labels. This in turn enables the classification algorithms to predict the decomposition vectors well. Additionally, low cardinality indicates there are only few dependencies between the labels. On a data set like *Scene*, where an average of only one label is positive per instance, the new algorithm does not predict well (see Figure 3.2). The Boolean decomposition of the matrix, when given only one label, does not improve the prediction. It still performs better than include labels, binary relevance and CLR, though.

Moreover, in all cases except *Emotions*, the classifier performs better than binary relevance, which can be seen as a single label approach, as all single label models are learned independently of each other. Only regarding recall, binary relevance outperforms MLC-BMaD. This is caused by the matrix multiplication step. The labels in $\mathcal{Y}'$ are predicted the same way as in BR, hence the predictions of these labels are approximately the same as the predictions of certain labels in $\mathcal{Y}$ when using BR. These predictions are multiplied with $\mathcal{M}$. Depending on the threshold in the decomposition, this matrix can be very sparse. In our experiments, the threshold was optimized and automatically set in most cases to a high value. Therefore, only labels which co-occur rather frequently are considered in this matrix. Multiplying with this matrix results in a matrix with more negative values than positives. Therefore, MLC-BMaD tends to predict more negative values when using a high threshold in the decomposition. Nevertheless, this could be avoided by setting a lower threshold in the decomposition step.

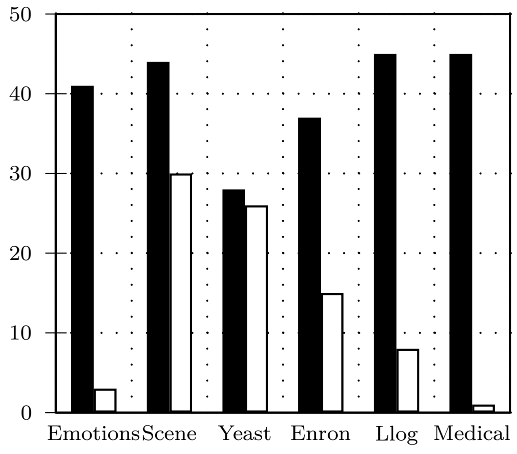


Figure 3.2: Pairwise comparison of MLC-BMaD over the data sets. The x-axis gives the data set. The y-axis indicates how often MLC-BMaD (black bars), or any other classifier (white bars) performs significantly better than the other on the given data set.

Table 3.3: Performance of different multi-label classifiers compared to MLC-BMaD. The symbol ◦ indicates a statistically significant increase of performance of the given method compared to MLC-BMaD regarding the given performance measure. The symbol • a statistically significant decrease, with the exception of Hamming Loss (statistically significant increase), with respect to the measure, a lower value represents an increased performance.

Measure	Data Set	MLC BMaD	MLkNN	LP	Stacking	BR	Include Labels	CLR	PLST
Accuracy	*Emotions*	0.515	0.324 •	0.446 •	0.534	0.523	0.442 •	0.458 •	0.495
	Scene	0.593	0.659 ◦	0.579	0.667 ◦	0.454 •	0.527 •	0.520 •	0.522 •
	Yeast	0.499	0.503	0.404 •	0.517 ◦	0.430 •	0.430 •	0.468 •	0.496
	Enron	0.396	0.304 •	0.331 •	0.299 •	0.234 •	0.398	0.411	0.257 •
	Llog	0.258	0.153 •	0.239	0.168 •	0.187 •	0.185	0.173 •	0.198 •
	Medical	0.737	0.543 •	0.730	0.541 •	0.315 •	0.736	0.724	0.593 •
FMeasure	*Emotions*	0.625	0.422 •	0.555 •	0.641	0.654	0.567 •	0.589 •	0.610
	Scene	0.630	0.681 ◦	0.603	0.688 ◦	0.599 •	0.582 •	0.588 •	0.552 •
	Yeast	0.637	0.638	0.533 •	0.649	0.567 •	0.583 •	0.625	0.6327
	Enron	0.531	0.425 •	0.447 •	0.418 •	0.439 •	0.538	0.555	0.392 •
	Llog	0.280	0.155 •	0.257	0.183 •	0.306	0.269	0.175 •	0.276
	Medical	0.782	0.577 •	0.766	0.603 •	0.364 •	0.782	0.771	0.690 •
Hamming Loss	*Emotions*	0.197	0.268 •	0.273 •	0.195	0.258 •	0.255 •	0.250 •	0.215
	Scene	0.107	0.090 ◦	0.147 •	0.088 ◦	0.240 •	0.136 •	0.138 •	0.117 •
	Yeast	0.200	0.197	0.282 •	0.195	0.300 •	0.251 •	0.221 •	0.204
	Enron	0.057	0.053 ◦	0.073 •	0.059	0.186 •	0.053 ◦	0.049 ◦	0.145 •
	Llog	0.018	0.016 ◦	0.027 •	0.031 •	0.236 •	0.017	0.016	0.107 •
	Medical	0.011	0.016 •	0.014 •	0.022 •	0.024 •	0.011	0.011	0.025 •
Precision	*Emotions*	0.656	0.512 •	0.560 •	0.658	0.571 •	0.560 •	0.552 •	0.628
	Scene	0.612	0.690 ◦	0.604	0.698 ◦	0.461 •	0.545 •	0.535 •	0.542 •
	Yeast	0.711	0.722	0.535 •	0.711	0.529 •	0.599 •	0.665 •	0.694
	Enron	0.570	0.540	0.453 •	0.488 •	0.329 •	0.601	0.643 ◦	0.316 •
	Llog	0.284	0.157 •	0.259	0.174 •	0.188 •	0.274	0.180 •	0.201 •
	Medical	0.783	0.593 •	0.781	0.590 •	0.368 •	0.772	0.759	0.618 •
Recall	*Emotions*	0.594	0.361 •	0.555	0.625	0.765 ◦	0.576	0.633	0.594
	Scene	0.647	0.680	0.604 •	0.678	0.855 •	0.624	0.653	0.564 •
	Yeast	0.577	0.572	0.531 •	0.597 ◦	0.610 ◦	0.568	0.589	0.582
	Enron	0.498	0.352 •	0.441 •	0.366 •	0.664 ◦	0.487	0.488	0.517
	Llog	0.276	0.152 •	0.254	0.194 •	0.839 ◦	0.281	0.174 •	0.439 ◦
	Medical	0.782	0.566 •	0.753	0.618 •	0.359 •	0.792	0.784	0.783
Subset Accuracy	*Emotions*	0.264	0.116 •	0.217	0.302	0.200 •	0.181 •	0.178 •	0.243 •
	Scene	0.526	0.613 ◦	0.539	0.629 ◦	0.175 •	0.424 •	0.397 •	0.467 •
	Yeast	0.146	0.170	0.128	0.189 ◦	0.096 •	0.062 •	0.094 •	0.145
	Enron	0.116	0.052 •	0.108	0.067 •	0.005 •	0.101	0.098	0.057 •
	Llog	0.220	0.145 •	0.205	0.154 •	0.141 •	0.214	0.162 •	0.153 •
	Medical	0.656	0.467 •	0.660	0.430 •	0.231 •	0.641	0.628	0.420 •

(continued)

Measure	Data Set	MLC BMaD	MLkNN	LP	Stacking	BR	Include Labels	CLR	PLST
Micro F1	*Emotions*	0.651	0.455 ●	0.561 ●	0.666	0.649	0.586 ●	0.613	0.630
	Scene	0.681	0.725 ○	0.592 ●	0.731 ○	0.558 ●	0.618 ●	0.625 ●	0.629 ●
	Yeast	0.632	0.636	0.531 ●	0.647 ○	0.547 ●	0.577 ●	0.616 ●	0.630
	Enron	0.513	0.458 ●	0.416 ●	0.440 ●	0.299 ●	0.526	0.553 ○	0.604 ●
	Llog	0.198	0.025 ●	0.124 ●	0.055 ●	0.096 ●	0.189	0.060 ●	0.091 ●
	Medical	0.799	0.652 ●	0.748 ●	0.602 ●	0.453 ●	0.799	0.793	0.630 ●
Macro F1	*Emotions*	0.611	0.356 ●	0.553 ●	0.645	0.643	0.575	0.604	0.616
	Scene	0.688	0.729 ○	0.737 ○	0.676	0.576 ●	0.629 ●	0.639 ●	0.635 ●
	Yeast	0.324	0.361 ○	0.374 ○	0.386 ○	0.449 ○	0.386 ○	0.387 ○	0.355 ○
	Enron	0.140	0.081 ●	0.138	0.128	0.166	0.152	0.137	0.146
	Llog	0.046	0.012 ●	0.044	0.029 ●	0.047	0.056	0.013 ●	0.051
	Medical	0.345	0.195 ●	0.339	0.217 ●	0.070 ●	0.363	0.347	0.352
Micro Precision	*Emotions*	0.723	0.626 ●	0.709	0.638 ●	0.565 ●	0.593 ●	0.592 ●	0.677
	Scene	0.731	0.806 ○	0.590 ●	0.814 ○	0.416 ●	0.607 ●	0.741	0.730
	Yeast	0.711	0.723	0.534 ●	0.713	0.503 ●	0.588 ●	0.651 ●	0.699
	Enron	0.570	0.660 ○	0.424 ●	0.554	0.197 ●	0.609 ○	0.675 ○	0.219 ●
	Llog	0.321	0.475	0.126 ●	0.052 ●	0.051 ●	0.253	0.580 ○	0.053 ●
	Medical	0.839	0.804	0.767 ●	0.609 ●	0.601 ●	0.819	0.814	0.533 ●
Macro Precision	*Emotions*	0.705	0.472 ●	0.560 ●	0.710	0.562 ●	0.586 ●	0.589 ●	0.671
	Scene	0.734	0.807 ○	0.604 ●	0.816 ○	0.446 ●	0.635 ●	0.624 ●	0.732
	Yeast	0.353	0.584 ○	0.377	0.560 ○	0.421 ○	0.406 ○	0.463 ○	0.464 ○
	Enron	0.147	0.134	0.145	0.192 ○	0.157	0.194 ○	0.223 ○	0.116
	Llog	0.070	0.035 ●	0.048	0.042 ●	0.027 ●	0.074	0.053	0.034 ●
	Medical	0.370	0.255 ●	0.359	0.233 ●	0.090 ●	0.374	0.376	0.330
Micro Recall	*Emotions*	0.593	0.360 ●	0.557	0.629	0.764 ○	0.582	0.636	0.590
	Scene	0.638	0.660	0.593 ●	0.664	0.847 ○	0.614	0.643	0.553 ●
	Yeast	0.570	0.568	0.529 ●	0.592 ○	0.601 ○	0.567	0.586	0.575
	Enron	0.467	0.352 ●	0.409 ●	0.365 ●	0.616 ○	0.464	0.468	0.498
	Llog	0.143	0.013 ●	0.124	0.057 ●	0.789 ○	0.147	0.032 ●	0.338 ○
	Medical	0.763	0.550 ●	0.731	0.596 ●	0.365 ●	0.779	0.772	0.771
Macro Recall	*Emotions*	0.575	0.322 ●	0.551	0.615	0.762 ○	0.572	0.627	0.581
	Scene	0.650	0.674	0.604 ●	0.676	0.845 ○	0.625	0.653	0.565 ●
	Yeast	0.321	0.332	0.373 ○	0.358 ○	0.526 ○	0.380 ○	0.370 ○	0.341 ○
	Enron	0.138	0.070 ●	0.140	0.117	0.357 ○	0.138	0.119	0.285 ○
	Llog	0.039	0.004 ●	0.047	0.036	0.404 ○	0.053	0.009 ●	0.181 ○
	Medical	0.338	0.174 ●	0.342	0.223 ●	0.065 ●	0.368	0.344	0.423 ○

Table 3.3 and Figure 3.3 also clearly show that MLC-BMaD outperforms the other algorithms in almost all cases on data sets with more labels: While all multi-label classifiers perform better compared to the Binary Relevance method on these data sets, MLC-BMaD is particularly good on these data sets, achieving an accuracy approximately twice as high as the accuracy of binary relevance for all these data sets. On the other hand, binary relevance seems to perform well regarding recall, both example-based, micro, or macro averaged recall. Compared to other methods, MLC-BMaD still performs

well regarding recall. In some cases it wins, in others it loses.

Further, if compared to the stacking method, MLC-BMaD also performs better on data sets with many labels. While on the smaller data sets, stacking outperforms MLC-BMaD, when having higher numbers of labels, MLC-BMaD outperforms stacking with respect to most performance measures. This is also expected as more labels produce a larger and more sophisticated association matrix for the decomposition and thus can produce a more meaningful decomposition.

Another advantage of MLC-BMaD is that it needs fewer classifiers on the base level and therefore less running time. The calculation of the decomposition matrices is not the most time consuming operation. Our tests on huge data sets showed that the method runs in less than 10 seconds, dominated by quadratic complexity (pairwise comparison of all labels in the decomposition step, see Algorithm 3.1). The same is true for the matrix multiplication in the final step of the prediction. The Boolean operations for the multiplication are not time critical. The base classifiers, on the other hand, can require quite some time to train. This is particularly true when using more elaborated methods, such as optimized support vector machines. In our experiments, we found the optimal number of decomposition matrix columns on bigger data sets to be at least 50% of the original size. Data sets with fewer labels need a higher ratio of decomposition matrix columns on the meta level. Still, on most data sets, the number can be reduced. Really

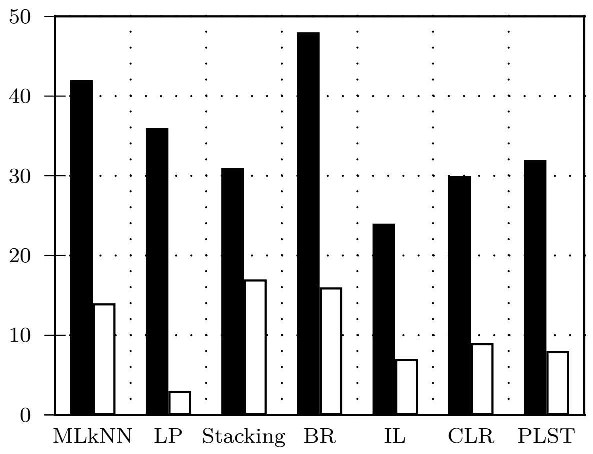


Figure 3.3: Pairwise comparison between MLC-BMaD and other multi-label classifiers. The x-axis gives the classifier to which MLC-BMaD is compared to. The y-axis indicates how often MLC-BMaD (black bars), or any other classifier (white bars) performs significantly better than the other on any data set.

small numbers of labels make the decomposition hard and not very useful: For *Emotions* and *Scene*, the six labels could not be reduced by much.

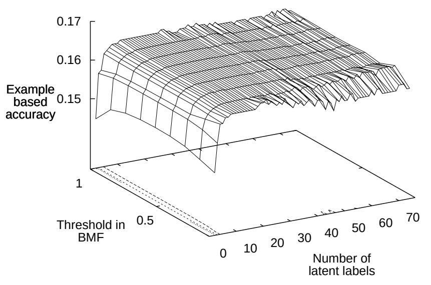
Comparing the performance on the different data sets with the cardinality in mind, the experiments show that MLC-BMaD achieves an advantage when using data sets with more labels and more dependencies among them. Moreover, the number of distinct label combinations has an impact on the choice of the classifier as well. Having many unique combinations makes some methods like label power set less efficient. For instance, the number of distinct label combinations and the cardinality on *Scene*, *Llog*, and *Medical*, is rather low, thus, on these data sets, label power set has a higher performance and reaches approximately the same level as MLC-BMaD. On the other hand, the number of distinct labels and the cardinality in *Emotions*, *Yeast* and *Enron?* is higher, thus the performance of label power set decreases and MLC-BMaD is better for most performance measures.

3.1.4.1 Influence of the Parameters on the Performance

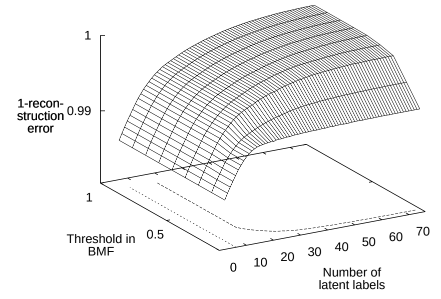
The Boolean matrix decomposition offers two parameters which have a big influence on the performance of the classifier. To visualize this, we performed additional experiments on the *Enron*, *Llog*, and *Medical* data sets. Using a *5-fold cross-validation* and random forests as a base classifiers, we evaluated every possible parameter setting. More precisely, we iterated k, the number of latent labels, from two to the number of labels of the data set, and τ, the threshold for the frequency in the decomposition, from 0.1 to 1.0. Figures 3.4a, 3.4c, and 3.4e give the result of this evaluation. The plots show a rather smooth function of the two parameters.

It should be noted that in these experiments, we used random forests as base classifiers. In these experiments, we wanted to evaluate the behavior of MLC-BMaD depending on the parameters of the BMD. We performed a *5-fold cross-validation* for every parameter setting and evaluated the results. This was done separately for three data sets. Hence, the performance shown in these figures differs from the one given in the other figures and tables. We chose random forest as we did not aim at precise results but at a more general view on the behavior depending on the parameters. This behavior should be less dependent on the base classifier. Random forests provided a faster alternative to SVMs.

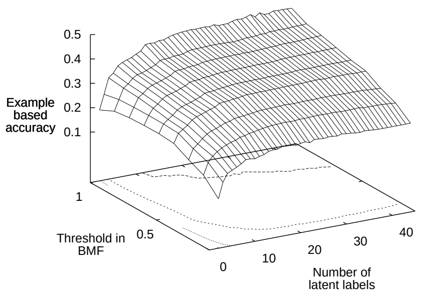
The value of τ has an influence on the result depending on the data set. On the language log data set, it does have a large impact. Some valleys and hills are visible in the area corresponding to low values of τ. However, larger values of τ seem to give a more stable performance. On the *Medical* data set, one can see that the performance



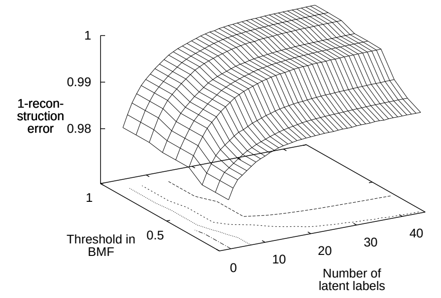
(a) Llog - performance of MLC-BMaD



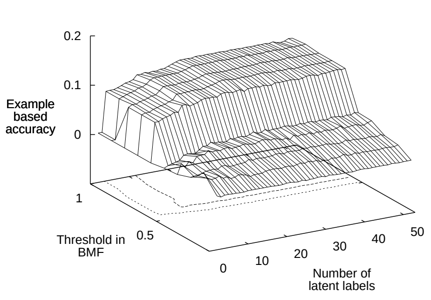
(b) Llog - Reconstruction error



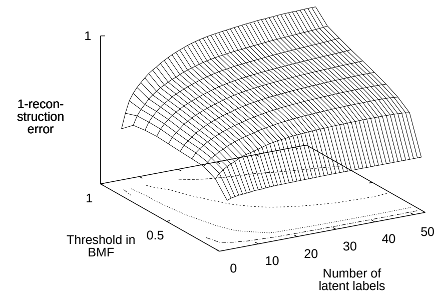
(c) Medical - performance of MLC-BMaD



(d) Medical - Reconstruction error



(e) Enron - performance of MLC-BMaD



(f) Enron - Reconstruction error

Figure 3.4: The performance (on the left side) of the MLC-BMaD classifier compared to the reconstruction error of the Boolean matrix decomposition, depending on the two parameters, number of latent labels and threshold for the matrix decomposition on three data sets. The reconstruction error is given as $1 -$ *reconstruction error*, to better visualize the relationship between reconstruction error and classifictaion performance. A *5-fold cross-validation* was used to calculate the example-based accuracy. As base classifier in this case we used random forests. Therefore, the accuracy differs from the one in the other experiments.

increases with higher τ. More precisely, the accuracy increases in two steps. A more complex relationship between the accuracy and τ is observed for the *Enron* data set. For lower τ values, the classifier cannot build meaningful models and the accuracy is close to 0. For higher values of τ, the performance increases again in one step. The plots show that it is beneficial to choose higher values of τ. With higher values of τ, only important candidates are selected for the decomposition. These can be seen as better representatives, being easier to learn. Also, the multiplication benefits from latent labels which occur more often in the original set.

On the other hand, the influence of k seems limited. The performance is low with k at low values. After a certain point, the performance changes only slightly for each data set. This makes sense as it is hard to decompose the label space into a matrix with too few latent labels and reproduce the original labels from that. Yet the stable performance is notable, no matter how high k is set. In other words, this shows that one can set k to a rather low value and get already a performance close to the optimum, while reducing running time and computing resources as the number of necessary classifiers is low. It should be noted that, while we chose to visualize the accuracy, the other performance measures behave similarly and converge to a similarly looking function.

To gain insight into the behavior of the algorithm regarding the reconstruction error in the BMD, we calculated the relative construction error for the three large data sets (*Medical*, *Llog* and *Enron*) for each parameter setting. The resulting plots (Figures 3.4b, 3.4d, and 3.4f) can be compared to the performance plots of the classifier. There seems to be a relation between the reconstruction error and the performance of the classifier. Nevertheless, the reconstruction error decreases with an increasing number of latent labels. The performance of the classifiers increases with the number of latent labels, but reaches a plateau at a certain point. The threshold in the matrix decomposition influences both values similarly. Steps in both functions are clearly visible. The reconstruction error can give some hints in which region the optimal setting for the classification lies. Yet the lowest reconstruction error is with the highest number of latent labels.

The plots also show that a method which can be used to optimize the two variables is hill climbing. There are rarely local minima or maxima which could confuse the hill climber. Additionally, it can terminate fast as after only few steps it can reach a stable plateau where the performance differences are minimal and terminate. This method seems to outperform the grid search used in the previous experiments, as it will find a close to optimum setting with far fewer internal evaluation steps.

3.1.4.2 Summary of the Results

To summarize, MLC-BMaD seems to be well-suited for complex multi-label data: in extreme cases, where data is almost single-label instead of multi-label, the performance of the stacking method is better, as can be seen for the *Scene* or *Emotions* data set. Still, MLC-BMaD outperforms other methods. When data exhibits at least some reasonable, yet possibly small label cardinality, like four in the case of the *Yeast* data set, then performance is already quite competitive. When data is really complex, like in the *Enron*, *Medical* or *Llog* case, the new method performs best and only calibrated label ranking and include labels can compete. It can handle data sets with many unique label combinations and strong dependencies between the labels better than comparable methods.

3.1.5 Conclusion

The main contribution of this work is the use of Boolean matrix decomposition for generating latent labels in the base level step as a substitution for the original labels. The latent labels identify and represent dependencies between the original labels in a compact manner. Their predicted values are multiplied with the second matrix from the decomposition process to obtain the final labels. The experimental evaluation showed that the new method works particularly well on data sets with a large number of labels and strong dependencies between them, outperforming most competing methods in that scenario. While the results are quite promising, there is room for further improvement. Future work could be to investigate alternative decomposition methods, both Boolean and numeric, and also try clustering labels for a more hierarchical multi-level decomposition approach.

3.2 Multi-Relational Learning

The previous section introduced a new multi-label classifier using Boolean matrix decomposition. In this section, we extend this approach by adding a new dimension to the data set, in order to provide a method for problems closely related to multi-label classification. More specifically, we deal with a special case of multi-relational learning, where the data is stored in exactly three relations. Two relations describe instances, the third describes a mapping between pairs of instances from the two relations. It can be understood as a multi-label extension where the target label is also described by a feature vector.

There exist two main approaches for these classification problems. The most common approach is to merge the relations into propositional form [10, 31, 79]. The propositional relation is used as an input for standard model learning algorithms. This approach approach generates data sets which quickly become too large to handle by standard algorithms[15]. Additionally, this approach is hard to evaluate, as it needs to be taken into account that all relations need to be split into training and test sets. As the relations are merged into one, an instance can contain a test instance from one relation and a training instance from another relation.

The second, more sophisticated approach, is to use multi-relational classification algorithms [25]. This approach leads to a larger complexity but does not use duplicate data entries. A large number of algorithms exist for this kind of problems. We propose a new approach in this category by exploiting the dependencies between the data sets using Boolean matrix decomposition to overcome the limitations of these approaches.

This new approach, called ClassFact, is a system with many classifiers to classify new pairs of instances. Large classifier systems seem to perform well on this kind of data sets. A disadvantage when evaluating these problems is the lack of publicly available data sets. Even though many scenarios exist which would benefit from ClassFact, most data sets are only available in a processed form, where it could be not possible to transform it back into their original form.

This section is organized as follows. First, we give an overview of ClassFact. Then, we point out similar approaches. Next, we define the notation and terminology in this section. Subsequently, we describe ClassFact and the experimental approach. Finally, we give and explain the experimental results and finish with a conclusion.

3.2.1 Multi-Relational Learning by Boolean Matrix Decomposition and Multi-Label Classification – ClassFact

In many practical applications [10, 31, 79], data are given in the form of three relations, one main relation containing the classifications of pairs of objects and two side relations describing the objects themselves. For instance, we might have the interactions of inhibitors (small molecules) with kinases [10], having inhibitors and kinases described in separate relations and the (binary) interactions between them in the main relation. Another example would be the growth inhibition of small molecules on tumor cell lines

15 Each instance of each relation is present multiple times in the resulting data set. Hence the size of the new data set is the size of the original ones multiplied by each other. This already leads to large data sets with rather small relations.

[79], with the small molecules and the cell lines described in separate tables, and growth inhibition information in the main relation. A third example is the gene regulation of yeast under certain conditions [31], depending on the state of regulators. Here, the conditions and the states of regulators are described in separate relations, whereas the states of the regulated genes are given in the main relation. Note that we still distinguish the two cases where class information is available for all pairs of objects and where class information is missing.[16]

Although we have a many-to-one relation from the main relation to the side relations and we could thus join the information from the whole data set into one large (sometimes huge) table, it often makes sense to model such problems as inductive logic programming (ILP) problems: First, it can be advantageous to use declarative language bias to control the order in which variables enter a model. Second, the language used to represent models could be more expressive and powerful than the representation of the data itself, for instance, by allowing the comparison of variables, if they are of the same type. Third, ILP representations are extensible by design. For instance, we tested the use of network information in addition to the three tables in previous work [31]. Fourth, it is far more space efficient to keep the information in three separate tables than to join everything into one, extremely redundant table.[17]

Most methods applied to such a problem will use the data as is and directly aim for the prediction of the class label, although there may be a lot of structure and redundancy in the class labels of pairs of objects: The classifications of pairs of objects will hardly be independent of the classifications of other pairs containing one or the other object from the pair. Also, and related to the former point, it may be possible to reduce the overall class information that needs to be predicted to reduce the risk of overfitting.

Therefore, we propose a procedure that first identifies structure in the class labels. The procedure, called ClassFact (classification factorization) in the following, uses a slightly different problem representation in the form of matrices and extracts mutual dependencies between the classes of the two partners, just considering associations in the class information, not considering features of the partners themselves. In this way, pseudo or latent classes of the partners are created, which capture much of the information from the original class labels and act like a proxy or abstraction of the original classes. In this way, the classification problem is simplified and also the amount of information to

16 Of course, in real-world data there will most likely be gaps in the class information, and the question is just to which extent class labels are missing. If class labels are missing to a lesser extent, methods for data imputation can be used to replace missing values in the training sets.

17 It is redundant because information about the two partners is repeated.

be predicted is reduced. In the final step of the procedure, the pseudo classes predicted for each of the partner objects are simply multiplied (in the sense of matrix multiplication, details to be explained in the subsequent section) to obtain the predictions for the original classes.

The problem definition applies to many scenarios. Nevertheless, there are not many publications for this type of problem. Commonly, the data is transformed to a propositional form. As work is usually published referring to this form, it is hard to link these publications to ClassFact. Possible domains are the prediction of connections between two nodes in a network. Given descriptive features for each node, a model could be learned on a known network and predict potential connections between two new nodes. Hence, a network could be learned using the information of a known network[18].

3.2.2 Related Work

There exist three main related publications related to ClassFact. Buchwald *et al.* [10] generated data sets of kinases and inhibitors and the interactions between them. The features of the inhibitors are chemical descriptors of the structures, generated using cheminformatics tools. The kinases are described using bioinformatics features and tools. The data set turned out to be hard to use in classification. The approach chosen by the authors was merging the data into its propositional form. For the evaluation, multiple approaches were used. In the soft case, a new kinase was evaluated against known inhibitors or vice versa. In this case, classifiers were able to predict the classes relatively reliably with good results. Nevertheless, this approach could have been improved by using a multi-label approach. The hard case for the evaluation was when the goal was to predict if an unknown inhibitor interacts with an unknown kinase. The classifier was learned on the data set excluding any information on one kinase and one inhibitor. The learned model was applied to the left out kinase and inhibitor. In this case, the approach could hardly beat random classification.

The next publication by Richter *et al.* [79] tries to build a model predicting if a small molecule inhibits the growth of tumor cell lines. The authors used Cubist[19] to build regression rules to predict the GI50[20] value of the cell line when exposed to the anti-cancer agents. Descriptors for the compounds were frequent substructures in the whole

18 For example, one could predict connections in social networks, i.e. if a person described by his or her given information, or features, is friend with another person, described by other features.

19 Cubist is available at `http://www.rulequest.com/cubist-info.html`.

20 The GI50 is the concentration of a compound under which the growth of the cell line is reduced by 50%.

data set. The method was evaluated using a hold-out evaluation. Hence, the scenario was similar to the soft evaluation of Buchwald *et al.* [10]. Based on this evaluation, the authors claim that they were able to learn well performing models using this data. However, one problem with this procedure is the size of this data set; 30.000 structures are tested against 37 cell lines. Hence, the propositional form becomes a huge data set. Learning models on this data set can consume a lot of time and computing resources and some algorithms might not be able to terminate.

Finally, Fröhler and Kramer [31] generated a data set to learn a model which predicts the regulation of a gene. The model uses, among other features, binding site information and the states of regulators. The data was modeled in multiple relations. Models were learned using TILDE [6] on these relations. The data set consisted of an expression state of genes under certain experimental conditions. Features were extracted for the genes and the conditions and stored in multiple relations. The expression state describes if the gene is over-, or under-expressed, or does not change under the conditions.

3.2.3 Notation and Terminology

We assume that a data set is given in terms of three relations, $target(K', K'', Y)$, $rel'(K', X'_1, \dots, X'_{m'})$ and $rel''(K'', X''_1, \dots, X''_{m''})$. The $target$ relation contains the classification Y of pairs of objects K' and K''. The classification of a pair is to be predicted from information from the other two relations, rel' and rel''. Variable K' is the key connecting tables $target$ and rel', and K'' the key connecting tables $target$ and rel''. The cardinality of rel' is n', the cardinality of rel'' is n'', and the cardinality of the relation $target$ is $n' \times n''$.[21] In other words, we assume that relation $target$ contains target values for each possible pair of objects from rel' and rel''. Furthermore, we assume that target variable Y is binary, i.e., it can take the values 0 or 1.

Given this problem setting, we can rephrase it in terms of matrices[22]: First, we are given two matrices corresponding to the relations rel' and rel''. The former corresponds to an $n' \times m'$ matrix $\mathcal{X}'$, the latter to an $n'' \times m''$ matrix $\mathcal{X}''$. Moreover, we have the $n' \times n''$ matrix of target values $\mathcal{Y}$ with $y_{ij} \in \{0, 1\}$. In other words, the target values are not arranged "vertically" for each pair as in the case of relations above, but in the form of a matrix. As part of the procedure described below, $\mathcal{Y}$ is factorized into an $n' \times k$

21 Variables n and m always denote the number of objects and the number of variables, respectively. Also note that we chose superscripts for identifying the two input relations, while subscripts are reserved for variable indices or indices of matrix elements in the remainder of the section.

22 Note that capital letters denote logical variables in the logical formulation and matrices in the matrix formulation of the problem.

matrix $\mathcal{Y}'$ and an $k \times n''$ matrix $\mathcal{Y}''^T$ such that $\mathcal{Y} \approx \mathcal{Y}' \cdot \mathcal{Y}''^T$. In this way, two multi-label classification problems arise, one for predicting $\mathcal{Y}'$ from $\mathcal{X}'$, and one for predicting $\mathcal{Y}''$ from $\mathcal{X}''$.

In terms of training and test sets, we are given $\mathcal{X}'_{Trg}$, $\mathcal{X}''_{Trg}$, and $\mathcal{Y}_{Trg}$ for training, and test set information $\mathcal{X}'_{Tst}$ and $\mathcal{X}''_{Tst}$. Then the task is to learn a model from $\mathcal{X}'_{Trg}$, $\mathcal{X}''_{Trg}$, and $\mathcal{Y}_{Trg}$, minimizing some loss function, and apply it to $\mathcal{X}'_{Tst}$ and $\mathcal{X}''_{Tst}$.

3.2.4 Classification Factorization – ClassFact

To give a detailed explanation of ClassFact, we first give an overview of the used algorithm for Boolean matrix decomposition. After this, we will explain the training of the model as well as the application to new examples. The notation is given in Figure 3.5.

3.2.4.1 Boolean Matrix Decomposition

In the experiments, we used the Boolean matrix decomposition approach proposed by Miettinen *et al.* [68]. This method first calculates associations between two class vectors of two examples in matrix $\mathcal{X}''$. In our case, associations are calculated between two classes. In the second step, candidate basis vectors are selected from the associations.

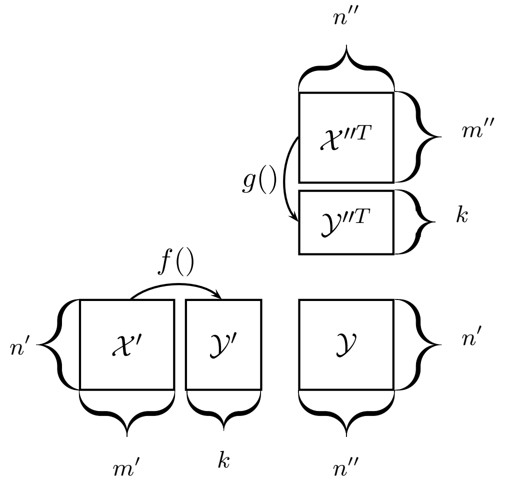


Figure 3.5: Notation used in this section. $\mathcal{X}'$ is the relation with the features of the first set of instances. $\mathcal{X}''$ is the relation with the features of the second set of instances. $\mathcal{Y}$ is the relation describing the class of two instance sets. Its values are 1 if there exists a mapping between the instances, 0 otherwise. $\mathcal{Y}'$ and $\mathcal{Y}''$ are the resulting relations produced by the Boolean matrix decomposition of $\mathcal{Y}$. The goal of the algorithm is to learn the functions $f()$ and $g()$ to predict $\mathcal{Y}'$ and $\mathcal{Y}''$ from $\mathcal{X}'$ and $\mathcal{X}''$. The final prediction is the product of $\mathcal{Y}'$ and $\mathcal{Y}''$.

Finally, a set of basis vectors is selected from the basis in a greedy manner. See Section 3.1.2.1 for more details and an example.

3.2.4.2 Learning the Model

The pseudocode algorithm to learn the model is given in Algorithm 3.4. In the first step, the matrix $\mathcal{Y}$ is decomposed into matrices $\mathcal{Y}'$ and $\mathcal{Y}''^T$. With each of these, a multi-label data set is composed with their corresponding feature matrix $\mathcal{X}'$ and $\mathcal{X}''$. For both data sets, a multi-label model is learned.

An potential mutli-label classifier for this task is binary relevance. This multi-label classifier learns one single label classifier $f_{1\ldots k}$ and $g_{1\ldots k}$ for each label. Binary relevance is the simplest learning scheme for multi-label classification, which does not take into account the dependencies among the labels. Using binary relevance, it would also be possible to use any ILP classifier and more complex relational background knowledge instead of $\mathcal{X}'$ and $\mathcal{X}''$, like for instance networks [31]. More advanced multi-label classifiers are particularly useful if k is too large and $\mathcal{Y}'$ still contains dependencies, or for the prediction of $\mathcal{Y}''$.

Due to the Boolean matrix decomposition, the upper matrix $\mathcal{Y}''^T$ contains the associations between the columns of the center matrix. Thus, the models $g_{1\ldots k}$ are learned to predict the associated columns of each column. The models $f_{1\ldots k}$ on the other hand can predict a compact representation of $\mathcal{Y}$. Single columns of this matrix which are important for all other columns can be predicted by $f_{1\ldots k}$.

3.2.4.3 Applying the Model to New Instances

To predict the class of a new set of instances (see Algorithm 3.5), we first predict the columns of the decomposed matrices $y'_{1\ldots k}$ and $y''_{1\ldots k}$. This is done by using the learned multi-label classifiers $f()$ and $g()$. The final predicted class of the new instance set is calculated by a Boolean multiplication of the predicted $\hat{y}'$ and $\hat{y}''$: $\hat{y} = (\hat{y}'_1 \wedge \hat{y}''_1) \vee (\hat{y}'_2 \wedge \hat{y}''_2) \vee \ldots \vee (\hat{y}'_k \wedge \hat{y}''_k)$. As the matrices $\mathcal{Y}'$ and $\mathcal{Y}''$ are predicted by the learned models, $\hat{y}'_{ij}$ and $\hat{y}''_{jl}$ with $i \in [1,k]$, $j \in [1,n']$, and $l \in [1,n'']$, are given by probabilities. Hence, the probabilities can be used in the multiplication step. This is done using the minimum of the values instead of the Boolean sum and the maximum of the values instead of the Boolean multiplication.

In the first step this algorithm predicts the associations of the new instance x''_i to k instances in the training set using $g_{1\ldots k}$. Next, $\mathcal{Y}'$ is predicted using $f_{1\ldots k}$. The final class of the test example is positive, if the predicted $\hat{y}''$ describes associations to the

Algorithm 3.4: Pseudocode for training the ClassFact classifier

```
Input: Data Set with features $\mathcal{X}'$,$\mathcal{X}''$ and class matrix $\mathcal{Y}$
Output: Sets of models $f$ and $g$
$(\mathcal{Y}', \mathcal{Y}''^T) \leftarrow Boolean\_Decomposition(\mathcal{Y})$
foreach $y'_i$ do
    /* train model $f_i$ for $y'_i$                                    */
    $\mathcal{D}'_i \leftarrow compose(\mathcal{X}',y'_i)$
    $f_i \leftarrow train(\mathcal{D}'_i)$
end
foreach $y''_i$ do
    /* train model $g_i$ for $y''_i$                                   */
    $\mathcal{D}''_i \leftarrow compose(\mathcal{X}'',y''_i)$
    $g_i \leftarrow train(\mathcal{D}''_i)$
end
```

Algorithm 3.5: Pseudocode for predicting the class of a new example using the ClassFact classifier

```
Input: Instances with features $x'$ and $x''$
Output: Predicted class $\hat{y}$
$\hat{y'} \leftarrow f(x')$
$\hat{y''} \leftarrow g(x'')$
$\hat{y} \leftarrow false$
foreach $i \in 1 \ldots k$ do
    /* $min(x,y)$ returns the minimum of $x$ and $y$, $max(x,y)$ returns the
       maximum of $x$ and $y$, $(\hat{y}_k = p)$ gives the predicted probability for
       value $p$ at $y_k$                                               */
    $\hat{y} \leftarrow min(\hat{y},max((\hat{y}'_k = 1), (\hat{y}''_k = 1))$
end
return $\hat{y}$
```

vectors of $\mathcal{Y}'$ in the training set and the predictions of $\hat{y}'$ predict at least one column of the compact representation in $\mathcal{Y}'$, which is predicted to occur frequently with one of the predicted $\hat{y}''$.

3.2.5 Experiments

To evaluate the performance of ClassFact, we joined the matrices into one and used 100-fold *cross-validation* to evaluate the method. The matrices $\mathcal{X}'$ and $\mathcal{X}''$ are both split into 10 folds each, and each combination of folds is used once for testing, the remaining for training. To estimate the performance, we calculated several performance measures.

As a base classifier for $f()$ and $g()$, we used random forests [9]. We implemented the method using multi-label classifiers from the Mulan library.

The orientation of the data sets has influence on the performance of ClassFact. Clearly, it matters which matrices are set to be $\mathcal{X}'$ and $\mathcal{Y}'$, and which $\mathcal{X}''$ and $\mathcal{Y}''$. There seems to be no simple way to estimate which orientation is the best, as it heavily depends where the decomposition method can find good associations. Thus, we tried both variants for each data set.

To gain the best setting for the parameters of the BMD, we used a repeated hill climbing optimization of the parameters with a *5-fold* internal *cross-validation.* For each training, we set the parameters randomly and evaluated these parameters including all settings around it. More precisely, we set the number k (rows of $\mathcal{Y}'$), as well as the threshold τ in the BMD. Subsequently, we set the parameters to the best setting evaluated in the round before. Next, we evaluate every setting in the grid around this setting. This procedure is repeated until the performance does not change more than 5% compared to the previous iteration. The whole procedure was repeated 5 times at 5 different start settings. The hill climbing was done for both matrix orientations independently. From both optimal settings in both orientations, the best was chosen and used to learn the final model.

3.2.5.1 Implementation

We implemented the algorithm as an extension to MLC-BMaD. We used Mulan to predict $\mathcal{Y}'$ and $\mathcal{Y}''$ from $\mathcal{X}'$ and $\mathcal{X}''$. The Boolean matrix decomposition was performed by a library reimplementing the method of Miettinen *et al.*[23] [68]. The advantage of this library is the capability to handle missing values. It is implemented in Java and hence can be included into Mulan and WEKA implementations.

3.2.5.2 Data Sets

To evaluate the performance of the method, we used three data sets already used in previous analysis [31, 10, 79]. The first data set by Fröhler and Kramer [31] describes the change in expression level of certain genes given an experimental condition. The experimental condition is additionally described by the change in expression of a set of

23 The library implements several methods to factorize a Boolean matrix. The process is split into three steps, each step can choose a range of methods. Methods can be arbitrarily combined to generate a BMD. One of the combinations is the approach by Miettinen *et al.* [68] We acknowledge the work of Andrey Tyukin, who was implementing this library.

Table 3.4: Characteristics of data sets used in the ClassFact experiments. P is the number of positive elements in $\mathcal{Y}$, N the number of negative elements.

Data Set	n'	m'	n''	m''	P	N
Kinase	19	150	177	451	894	2469
Gene strict	156	52	1410	305	194715	1798
Gene non strict	156	52	1410	305	172463	24050
NCI	1000	18	37	15	7184	8100

regulators in $\mathcal{X}''$. The genes are described by the mapping to certain transcription factor binding sites in $\mathcal{X}'$. The matrix $\mathcal{Y}$ contains the genes as rows and the experimental conditions as columns. The target value is a numeric value indicating the level and direction of change, i.e., it is a positive value if the expression level is increased, a negative value if the level is decreased and 0 if the level does not change. The numeric value gives the magnitude of the change.

For the experiments, we transformed the target values according to different settings. In the first setting, we transformed each positive and negative value to 1, and kept 0 the same. This scenario generates models predicting if there is any change in the expression level. The disadvantage of this strict setting is the high number of positives and low number of negatives (see Table 3.4, *Gene strict*). Hence, we added a second, less strict setting, using values between -0.1 and 0.1 as 0 and the remaining as 1. This resulted in a bit larger negative class (*Gene non strict* in Table 3.4). We evaluated both sets, as it seems easier and more natural to differentiate between no change and change, but from the machine learning perspective it might be easier and more natural to differentiate between an approximate no change and a clear change.

The second data set by Buchwald *et al.* [10] (*Kinase*) contains the interactions of inhibitors with kinases. The rows of $\mathcal{X}'$ contain the kinases, the columns the features computed by bioinformatics tools. The matrix $\mathcal{X}''$ contains the inhibitors as rows and several chemical descriptors of the structure as columns. Matrix $\mathcal{Y}$ describes if there is an interaction between the kinases in the rows and the inhibitors in the columns.

Finally, the last data set by Richter *et al.* [79] (*NCI*) captures the concentration of a certain substance at which the growth of a cancer cell line is reduced by 50%. $\mathcal{X}'$ contains the cell lines in the rows, described by the similarity to other cell lines and tissue types. The substances are in the rows of $\mathcal{X}''$, described by the similarity to other anti-cancer agents determined by substructural descriptors in the columns. $\mathcal{Y}$ gives the discretized growth inhibition value.

3.2.5.3 Evaluation Measures

Since the classifier returns probabilities for each prediction, standard machine learning measures can be used. We generated precision-recall plots and ROC curves. precision-recall plots have the recall on the x-axis and the precision on the y-axis. Recall is the fraction of true positives of all positives (see Equation 3.4). Precision is the the fraction of true positives of all predicted positives (see Equation 3.5). The plots show the trade off between the two measures. Each point gives recall and precision using a certain threshold for the classfication of an instance belonging to the positive class. If the threshold is low, all instances are classified as positives, hence all positives are recognized correctly, while all negatives are classified as positives as well. Thus, recall is high and precision is low. This changes if the threshold is raised, recall decreases and precision increases. Note that, although the plots cannot be interpolated [15], we nevertheless connected the points for better visualization.

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.5}$$

ROC curves plot the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis. TPR is the same as recall and gives the fraction of correctly classified positive examples over all positive examples (see Equation 3.6). FPR gives the fraction of positively classified negative examples over all negative examples (see Equation 3.7). Similar to recall and precision, these values represent a trade off when choosing the optimum threshold for the classification. If a low threshold is chosen, many or all examples are classified as positive, hence TPR is high. At the same time, many negative examples are classified incorrectly as positive, hence FPR is high. If a high threshold is chosen, almost none of the examples is classified as positive, therefore FPR is low. Also, TPR is low, as many positives are missed.

$$TPR = \frac{TP}{TP+FN} \tag{3.6}$$

$$FPR = \frac{FP}{FP+TN} \tag{3.7}$$

3.2.5.4 Results

The main problem when evaluating ClassFact is the low quality of the used data sets[24]. ClassFact as well as the baseline method are not able to generate well performing models, the average performance is close to random. This is in a way expected as we use simplified versions of the data sets [10, 79, 31].

Nevertheless, Table 3.5 clearly shows an improvement when using ClassFact. In the case of *Kinase*, the baseline method was not able to produce a model predicting anything better than random. ClassFact found a signal in the data and was able to generate a model which performs clearly better than random. The difference between the classifiers in terms of area under the ROC curve is rather big, ClassFact performs 20% better than the baseline method. In the case of the *NCI* data set, neither of the classifiers can outperform the other one. The baseline method performs slightly better. This can be due to the adaptation of the data set to a classification task. In its original form, the goal was to predict a numeric value, the concentration of a substance reducing the cancer cell line growth. In our case, we discretized the data set to an almost even class distribution and predicted the resulting class. In the two scenarios of the *Gene* data set, ClassFact can slightly outperform the baseline method.

It should be noted that ClassFact performs better than the baseline method even on a very skewed class distribution. The strict case of *Gene* performs better than the milder

Table 3.5: Area under the ROC curve of the baseline method and ClassFact on the data sets. Bold marked are the highest values per row. In the majority of cases, ClassFact can improve the performance of the baseline method. Even with a skewed class distribution (*Gene strict*), it can outperform the baseline. Nevertheless, all three data sets cannot be used very well for prediction, the performance is close to random, which confirms the assumption that the data sets are very hard for prediction purposes. Still, ClassFact can find a signal in the data where the baseline method does not work.

Data Set	Baseline	ClassFact
Kinase	0.438	**0.561**
Gene strict	0.475	**0.538**
Gene non strict	0.499	**0.516**
NCI	**0.514**	0.512

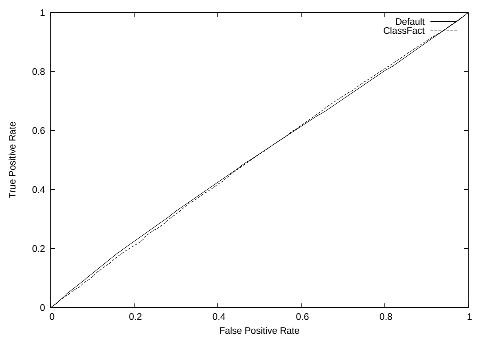
24 This is due to the low number of available data sets. Even in their original publications, the generated models only exhibited average performance using sophisticated methods. For the evaluation, we used random forest classifiers without elaborated parameter tuning.

setting with a threshold for the change in growth. The strict case has only 0.8% of the instances in the negative class (see Table 3.4). Still, even at that level, ClassFact is able to find at least some discrimination between the two classes. Most classifiers suffer from such distributions, it seems that ClassFact is more immune to these settings. The worse performance of ClassFact when using the milder setting can be attributed to a weaker signal in the data to distinguish between two changes at a low level compared to a stronger signal to distinguish between no change at all and some change. Nevertheless, ClassFact does not depend on uneven class distribution as Kinase has a rather balanced distribution.

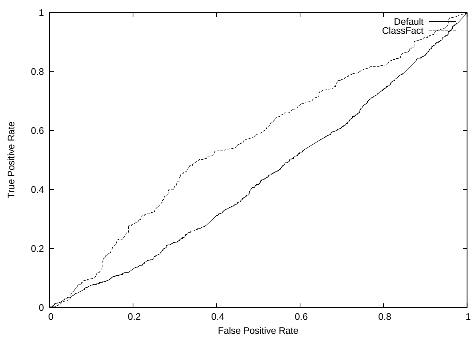
Regarding the random performance on the *NCI* data set, it should be noted that this data set is a reduced version from the one used in Fröhler *et al.* [31]. As we randomly selected 1000 instances in $\mathcal{X}'$ and $\mathcal{Y}$, there might be some information loss in the data, and hence, it might be harder to learn meaningful models. Still, the baseline method does not perform substantially better than ClassFact, they perform on a similar level.

The ROC curves (see Figures 3.6) show in which cases one classifier outperforms the other. They show that the baseline method never has a higher true positive rate and lower false positive rate at the same time. This is strongly visible for the *Kinase* data set, but also the *Gene* data sets show this. On the *NCI* data set, one can see that both classifiers perform the same, not predicting anything but random. Nevertheless, the other plots show that ClassFact can find something in the data. The improvement is not random, it can be seen that ClassFact finds some signal in the data at certain points. An interesting shape of the ROC curve can be seen on the strict *Gene* data set. The data rows by the baseline method and ClassFact are on distance on the lower levels, in the middle the almost meet while on higher levels, ClassFact again improves the prediction. This implies that ClassFact has more extreme probabilities as predictions than the baseline method. The probabilities are either high or low, hence giving stronger signals in these areas. This leads to a better prediction, as ClassFact gives the correct prediction more often.

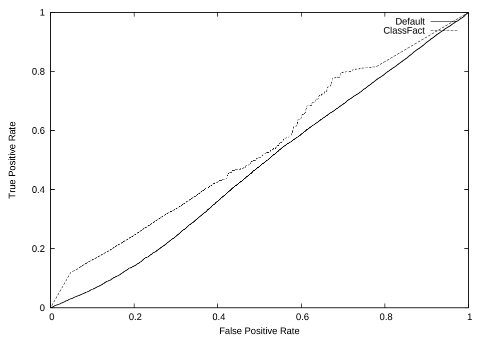
The precision-recall plots in Figure 3.7 show a similar pattern. Precision is almost constant over all thresholds for the baseline method on all data sets, yet, ClassFact can improve the precision when using a higher threshold. Again, this is most evident in the case of *Kinase*. Having an almost constant precision means that the number of true positives does not change a lot, or the number of false positives is too high throughout all thresholds. ClassFact can improve the precision on all data sets with the exception of the *NCI* data set, ClassFact is good at reducing the number of false negatives, i.e., the positives get predicted more reliably using ClassFact.



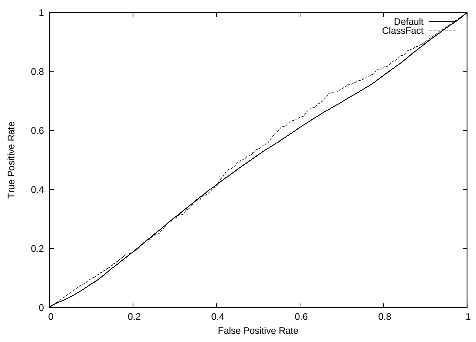
(a) NCI



(b) Kinase



(c) Gene strict



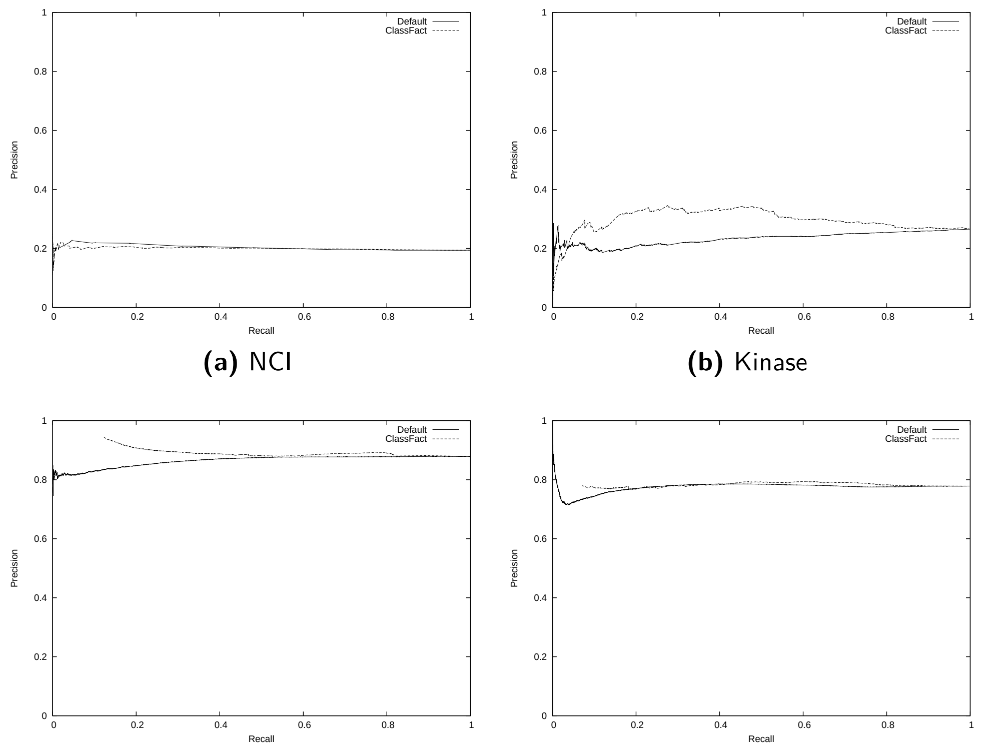
(d) Gene non strict

Figure 3.6: ROC curves for ClassFact and the baseline method on all data sets.

3.2.6 Conclusion

This section proposed a new method for a frequent type of relational learning problem, i.e., predicting the class of a pair of objects. Problems of this type are abundant in the areas of computational biology, respectively bioinformatics and cheminformatics, where the task is often to predict the effect of a change of parameters on a system, e.g., how a biological system reacts to a drug or toxicant. The availability of systematically collected data for pairs of objects in these areas will further increase the need for algorithms analyzing this kind of data.

Having said that, evaluating this approach is difficult, as there rarely exist data sets in that form yet. Although problems of a similar structure exist, the common approach is to transform them to single-label, single-relational data sets and use simple classification algorithms. Hence, the question arises whether it would possible to extract the original relations from transformed data sets. A method transforming the data sets back to its



(a) NCI **(b)** Kinase **(c)** Gene strict **(d)** Gene non strict

Figure 3.7: Precision-recall plots for ClassFact and the baseline method on all data sets.

original state would enable to apply the proposed algorithm to these problems. Additionally, it might be possible to extract unknown patterns from data and transform parts of standard data sets to be usable by ClassFact. This might find and exploit relations in the data sets which were not known and used before.

The approach works by Boolean matrix decomposition and multi-label classification, and thus provides a new angle on relational learning in such a setting. Experiments on three different real-world problems show that the method already produces results better than the baseline approach, which is the most widely used approach. It can be understood as an extension to MLC-BMaD, as introduced in the previous section.

3.3 Conclusion

In this chapter, we introduced Boolean Matrix decomposition for use in large classifier systems. This is particular useful, as it reduces the number of classification tasks and exploits dependencies between the tasks. In the case of multi-label classification, it exploits the dependencies between labels and strongly reduces the number of learning tasks. When handling data sets with a big number of labels, this helps improving the running time and performance.

Subsequently, we extended the multi-label task to a multi-relational problem by adding descriptions to the labels and, based on this information, predicted new labels for new instances. This is a special case of multi-relational learning. However, the evaluation is hard, as most of these problems get transformed into propositional form, hence the structure and information for this task is lost.

The presented results are promising and show that Boolean matrix decomposition can be useful in large classifier systems. Future work could aim to extend the proposed algorithms further to handle propositional data sets by reconstructing their multi-relational structure. Additionally, other factorization methods, e.g., autoencoders instead of Boolean matrix decomposition, might be used.

In the following chapters, we will focus on the applications of large classifier systems in the life sciences. First, we will generate models to predict the environmental fate of chemicals, subsequently we use large classifier systems to predict the toxic effects of chemicals. These domains show an application for the methods introduced in this chapter.

CHAPTER 4

Biodegradation Pathway Prediction

In silico methods to predict products and pathways of microbial biotransformations of chemical substances are increasingly sought due to rapidly growing data requirements for regulatory chemical risk assessment at the European (cf. REACH [75]) and global level. Existing methods for the prediction of biotransformation products and pathways can be categorized as either knowledge-based or machine learning-based approaches. Each of the two approaches has its strengths and weaknesses. Knowledge-based approaches such as METEOR for the prediction of mammalian metabolism [36] or the University of Minnesota Pathway Prediction System (UM-PPS) for microbial biodegradation [43] take into account expert knowledge on the basis of sets of transformation rules. However, they run the risk of including potentially overly-general, incomplete, or inconsistent rules. In contrast, machine learning approaches produce accurate probability estimates on the basis of empirical data, but often lack the ability to incorporate prior domain knowledge. Also, recent machine learning approaches for biotransformation prediction only predict quite general classes (e.g., whether a compound plays a role in central metabolism [35] or whether it is the substrate of some broad reaction class, e.g. oxidoreductase-catalyzed reactions [70]).

The goal of this chapter is to combine the two approaches: We assume a given set of biotransformation rules and learn the probability of transformation products proposed by the rules from known, experimentally elucidated biodegradation pathways. Only two comparable systems exist so far: META [57] which is similar in spirit, but uses a less advanced problem formulation and machine learning approach than the one presented here, and CATABOL [19], the only rule-based method explicitly aiming for probability estimates. However, the CATABOL system works with a fixed pathway structure for training, which is different from the approach presented here, working on the basis of individual rules (for a detailed discussion, see Section 4.6).

Rule-based systems, such as UM-PPS work on the basis of rules that are generalizations and abstractions of known reactions, in the case of UM-PPS, its underlying Biocatalysis/Biodegradation Database (UM-BBD) [27]. UM-BBD is a manually curated compilation of over 200, experimentally elucidated microbial biotransformation pathways, encompassing enzymatic reactions for roughly 1,000 parent compounds and intermediates. If certain functional groups of a query substrate match with any of the biotransformation rules in UM-PPS, then its structure is transformed into one or several products according to the rule(s). These rules are typically fairly general, either to cover all known reactions, or because there is not enough information known to restrict them. As a consequence, UM-PPS produces a large number of possible reaction products, especially when used to predict several subsequent generations of transformation products. This combinatorial explosion is a phenomenon also known from other rule-based systems and approaches. It is particularly aggravated for the structurally more complex contaminants of current concern, e.g., pesticides, biocides or pharmaceuticals. Potential users of such a system, such as environmental microbiologists, risk assessors, and analytical chemists, are overwhelmed by the number of possible products, and find it hard to identify the most plausible products.

In an effort to restrict combinatorial explosion, some of the knowledge-based approaches to metabolic prediction employ what is called *relative reasoning* [11]. In relative reasoning, the possibility to apply a rule depends on the presence of other applicable rules. Practically, this requires additional rules for the prioritization of rules and the resolution of conflicts. These meta-rules, or relative reasoning rules, express that some reactions take priority over others, and vice versa, that some reactions only occur if others are not possible.

Relative reasoning rules have recently been derived automatically for the set of UM-PPS biotransformation rules and have been successfully implemented into the working UM-PPS [28]. However, although reductions in the number of predicted products in one prediction step of about 20% were achieved, the selectivity (precision) of UM-PPS still remained rather low, at about 16-18%. Thus, the question remains how we can further refine the process of selecting and accepting those rules that most likely lead to observed products, when a set of rules applies to the structure of a given substrate.

In this chapter, we propose a solution that transfers the idea of relative reasoning to a machine learning setting, to further improve the system's selectivity (precision). Rule probabilities are to be estimated such that they depend not only on all other rules that are applicable, but also on the structure of the substrate. The priorities are learned statistically from data on known biodegradation pathways. In our solution, one classifier is learned for each rule, generalizing over the molecular substructures of the substrate and the "activation patterns" of the rules as given by the set of all other rules that are triggered by the same substrate.

Given the availability of such probabilistic classifiers, the decision to accept a product or not can be made dependent on a probability threshold: The application of individual rules can be tuned such that only transformations with a probability above a certain threshold are accepted. In this way, one can also control the generality of whole rule sets and the overall number of products. Thus, it is easy to address the fundamental trade-off between the completeness and the accuracy of predictions. In technical terms, we can analyze the performance of both individual rules and the whole system in precision-recall space, and visualize their performance in two-dimensional plots. Moreover, it is possible to explicitly choose a suitable point in precision-recall space by setting the probability threshold for accepting a rule to a certain level.

4.1 A Hybrid Knowledge and Machine Learning-Based Approach

To illustrate the problem and the proposed solution, we start with an example shown in Figure 4.1a: Given a new compound c_{new}, several rules of the UM-PPS are applicable and

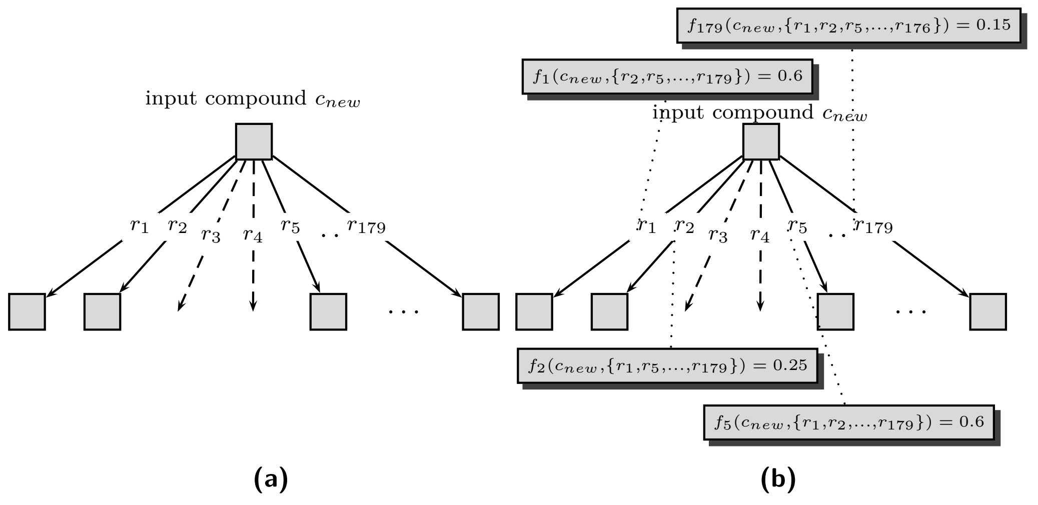
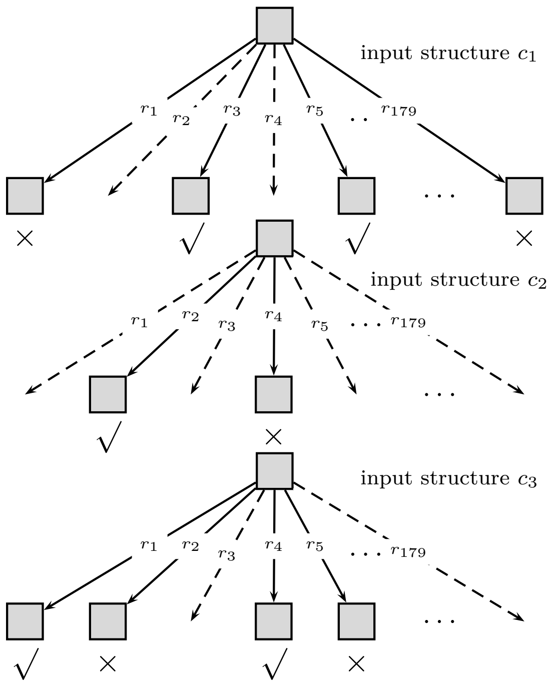

input compound c_{new}
r_1 r_2 r_3 r_4 r_5 r_{179}
...
(a)
$f_{179}(c_{new},\{r_1,r_2,r_5,\ldots,r_{176}\}) = 0.15$
$f_1(c_{new},\{r_2,r_5,\ldots,r_{179}\}) = 0.6$
input compound c_{new}
r_1 r_2 r_3 r_4 r_5 r_{179}
...
$f_2(c_{new},\{r_1,r_5,\ldots,r_{179}\}) = 0.25$
$f_5(c_{new},\{r_1,r_2,\ldots,r_{179}\}) = 0.6$
(b)

Figure 4.1: (a) indicates the rules applicable to an input compound c_{new} by solid arrows, rules not triggered are given by dashed arrows. (b) illustrates the use of one classifier for each rule to determine the probability of obtaining a valid product, depending on the structure and other applicable rules.



input structure c_1
r_1 r_2 r_3 r_4 r_5 ... r_{179}
input structure c_2
r_1 r_2 r_3 r_4 r_5 ... r_{179}
input structure c_3
r_1 r_2 r_3 r_4 r_5 ... r_{179}

(a)

	s_1	...	s_{547}	r_2	r_3	r_4	r_5	...	r_{179}	$y = r_1$
c_1	+1	...	0	0	+1	0	+1	...	+1	0
c_3	+1	...	+1	+1	0	+1	+1	...	0	+1
...	...	...	...	...	...	...	...	...	...	...
c_{718}	0	...	0	+1	0	0	+1	...	0	0

(b)

	s_1	...	s_{547}	r_1	r_3	r_4	r_5	...	r_{179}	$y = r_2$
c_2	0	...	+1	0	0	+1	0	...	0	+1
c_3	+1	...	+1	+1	0	+1	+1	...	0	0
...	...	...	...	...	...	...	...	...	...	...
c_{718}	0	...	0	+1	0	0	+1	...	0	+1

(c)

Figure 4.2: Figure (a) gives examples for the construction of two training sets, one for f_1 (b) and one for f_2 (c). Observed products are indicated by a check mark, spurious products are indicated by a cross.

suggest possible transformation products. In the example, a subset of rules r_1, r_2, r_5, etc. triggers for the given input structure. In the illustration, triggering rules are indicated by solid arrows, rules not triggering by dashed arrows. As mentioned above, the problem is that the rules of the system are overly general, i.e., they suggest a wide range of possible products, many of them false positives.

To restrict the number of possible products, it would be desirable to score the proposed transformations by estimated probabilities. In this way, it would also be possible to tune the number of products depending on a user-defined threshold: If the estimated probability of a transformation exceeds a threshold, it is accepted, otherwise it is discarded. The probability for each rule r_i is estimated by a corresponding function f_i. Function f_i tells us how likely a transformation suggested by rule r_i is, depending on the structure of the input compound and all other triggering rules. This is illustrated in Figure 4.1b: Function f_1 estimates that the probability of obtaining a correct product from applying r_1 to substrate c_{new} is 0.6, given the molecular structure of c_{new} and the other rules applicable $(r_2, r_5, ..., r_{179})$. The dependency of the decision on all other transformation options reflects that, under certain conditions, one reaction should be given priority over another. If the cut-off was set to 0.5 in the example, we would only accept the transformations proposed by r_1 and r_5.

The problem is of course to derive suitable probability scores. In this chapter, the solution is based on a training set of examples and machine learning. In Figure 4.2a, a sample of three compounds from a hypothetical training database is shown. For the three training compounds, we assume that we not only know which rules are applicable, but also which rule applications lead to observed products. In the figure, the observed transformation products are indicated by a check mark, whereas the spurious products are indicated by a cross. Given this information, it is possible to learn under which conditions the suggested product of a transformation rule can actually be observed. As a classifier is only needed when a rule triggers, the training set for a rule also includes only those compounds for which the rule suggests a product. Figures 4.2b and 4.2c show two training sets constructed from the three training compounds c_1 to c_3, one for rule r_1 (upper table) and one for rule r_2 (lower table). The first group of features $(s_1, ..., s_m)$ is a fingerprint-based representation of the structure of the input compound. The second group of features (all $r_1, ..., r_{179}$ except the rule for which the classifier is built) indicates which other rules are applicable to the compound: A feature is set to $+1$, if the corresponding rule fires, and 0, otherwise. As explained above, the training set for f_1 does not contain an entry for c_2, because rule r_1 is not applicable to that compound. Similarly, c_1 is not listed in the training set for f_2, because rule r_2 cannot be applied. Also note that c_3 is a positive example for f_1, while it is a negative example for f_2. Given such training sets, any machine learning algorithm for classification can be applied to induce a mapping from the structural and rule descriptors to the target variable, i.e., whether a rule generates an observed product.

To be more precise (amongst others, to enable reproducibility), we introduce some notation: In the following, C denotes the set of compounds c_i, and R the set of rules r_j. $triggers(r_j, c_i)$ is a predicate indicating whether r_j triggers on compound c_i. $observed(r_j, c_i)$ is a predicate indicating that rule r_j fires and provides an observed degradation product. For instance, we have the following list of facts for c_1 and c_2, and r_1 to r_3 from Figure 4.2a:

$$triggers(r_1, c_1). \tag{4.1}$$
$$triggers(r_3, c_1).\ observed(r_3, c_1). \tag{4.2}$$
$$triggers(r_5, c_1).\ observed(r_5, c_1). \tag{4.3}$$
$$triggers(r_2, c_2).\ observed(r_2, c_2). \tag{4.4}$$
$$triggers(r_4, c_2). \tag{4.5}$$

Finally, S denotes the set of molecular substructures s_l, and predicate $occurs(s_l, c_i)$ checks the occurrence of a substructure s_l in a compound c_i.

To prepare for training, we need two transformation operators, one for the construction of individual examples, and one for the construction of whole training sets. The first one, $\tau_{instance}$, is defined as follows:

$$\begin{aligned}\tau_{instance}(c_i, k) = x_i,\ such\ that&\\ &x_{i,j} = occurs(s_j, c_i)\ for\ 1 \leq j \leq |S| \wedge\\ &x_{i,j} = triggers(r_j - |S|, c_i)\ for\ |S| < j \leq |S| + k - 1 \wedge\\ &x_{i,j} = triggers(r_j - |S| + 1, c_i)\ for\ |S| + k - 1 < j \leq |R| + |S| - 1\end{aligned} \tag{4.6}$$

This means that operator $\tau_{instance}(c_i, k)$ constructs the description of an individual example without its class information. It takes a compound c_i and constructs a feature vector (see the example above), taking into account substructures and applicable rules. Parameter k is used to exclude the information for the k-th rule, which is convenient for our purposes, because it constitutes the target for training. With $\tau_{instance}(c_i, k)$, we are ready to define a transformation operator generating a training or test set for rule k from a given set of compounds C: τ_{set} takes a compound c_i from C and checks whether rule k triggers. Only if this is the case, a training example (x_i, y_i) is constructed:

$$\tau_{set}(C, k) = \{(x_i, y_i) | c_i \in C \land triggers(r_k, c_i) \land x_i = \tau_{instance}(c_i, k) \land \\ y_i = 1\, if\ observed(r_k, c_i), y_i = 0\ otherwise\} \quad (4.7)$$

Algorithm 4.1: Pseudocode for training and testing classifiers for biotransformation rules

```
Input: Training data $\mathcal{D}_{Trg}$, rules $r$, test instance $c_{new}$, threshold $\theta$
Output: Predicted products of $c_{new}$
/* training one classifier per rule                                    */
foreach rule $r_k$ do
    $\mathcal{D}^k_{Trg} \leftarrow \tau_{set}(C_{Trg}, k)$ $f_k \leftarrow train(\mathcal{D}^k_{Trg})$
end
/* testing for a new test compound $c_{new}$                             */
/* the cut-off for acceptance is given by parameter $\theta$             */
foreach rule $r_k$ do
    if $triggers(r_k, c_{new})$ then
        if $f_k(\tau_{instance}(c_{new}, k)) > \theta$ then
            classify as “product of $k$”
        else
            classify as “no product of $k$”
        end
    end
end
```

In the example above, $\tau_{set}(\{c_1, c_2, c_3..., c_{718}\}, 1)$ gives us the training set for classifier f_1 shown in the table of Figure 4.2b, $\tau_{set}(\{c_1, c_2, c_3..., c_{718}\}, 2)$ gives us the training set for f_2 in the table of Figure 4.2c. A training procedure *train* returns the classifiers needed for the restriction of the rules based on such training sets. As already indicated above, classifiers are represented as functions f_j returning class probability estimates for given examples.

Given those preliminaries, we can explain how training and testing is performed and how it is embedded into the working system (see Algorithm 4.1 for the pseudocode). In the training phase, a classifier is trained for each rule in turn. In the testing phase, we first obtain a list of rules applicable to each test compound using the UM-PPS. If a rule triggers, we apply the rule's classifier to the instance, where information from the molecular structure and all competing rules is taken into account to obtain a probability estimate. If this estimate exceeds a threshold θ, the product suggested by rule r_k is accepted, otherwise, the proposed transformation is rejected.

4.2 Using Multi-Label Classifiers and Extended Encoding

Clearly, there are dependencies between the rules. If one rule is correctly applied to a structure, other rules might not be applicable applied or supported by this transformation. Nevertheless, the method proposed above does not exploit the dependencies for the training. Multi-label classifiers can exploit the dependencies between multiple binary target values. We decided to use ensembles of classifier chains (see Section 2.1.4.3) and MLC-BMaD (see Section 3.1.2) on the data set.

In addition, we changed the encoding of the target information to improve the classification and make the problem more suitable for multi-label classifiers. An overview of the mapping is given in Table 4.1, an example data set is given in Table 4.2. The first set of labels for the classifiers was set to binary values describing if the corresponding rule was triggered correctly or not. If a rule was not triggered at all, this value was set to be missing. Additionally, the second set of labels provided information about the activation (triggering) of rules: if the rule is incorrectly triggered or not triggered (negative label), or triggered correctly (positive label). While this encoding of the information might seem redundant and not intuitive, multi-label algorithms benefit from this redundant information. The second set of labels describes if a rule should be used for the compound or not, no matter whether they are triggered or not. The latter means

Table 4.1: Extended encoding for multi-label classification. "?" indicates a missing value, which is ignored by the classifiers and evaluation. The data is translated into three new attributes, two are used as labels as they store the target information if a rule is triggered correctly. The third attribute is used as feature, used for the predictions. For the final prediction, a combination of the probabilities of label 1 and label 2 is used. However, for the evaluation, only label 1 is used.

	correctly triggered	incorrectly triggered	not triggered
label 1 (λ_i) / correctly triggered	1	0	?
label 2 (λ'_i)/ known product	1	0	0
feature (x_i)/ triggered	1	1	0

that the environmental fate strictly is predicted on a machine learning basis, leaving the expert-based knowledge aside. The expert-based knowledge is given by the features telling if a rule is triggered or not. For the final prediction, we used a combination of the two labels per rule. For each rule, we combined the predicted probability of each set using the mean of the predictions. For the evaluation process, we only used the label set that tells if a label is correctly triggered. Only from them, evaluation measures are calculated, which is the same values on which the single-label approach is evaluated.

4.3 Data and Implementation

It is possible to validate the procedure described above by running a cross-validation over the compounds of the UM-BBD database. The running time can be optimized considerably if the predicates *triggered*, *observed*, and *occurs* are pre-computed once for all compounds and stored for later use. In our implementation, we pre-computed a $|C| \times |R|$ table indicating the rules' behavior on the UM-BBD compounds: The value $+1$ of an entry encodes that a rule is triggered and produces an observed product, 0 encodes that a rule is triggered but the product is not observed, and -1 encodes that the rule is not triggered for a given compound. Simple database and arithmetic operations can then be applied to extract training and test sets, e.g., for (*leave-one-out*) *cross-validation.* Additionally, we learned the classifiers on the complete data set and tested the proposed approach on an external validation set of 25 xenobiotics (pesticides), which was also used in previous work [28].[25] Pesticide biodegradation data is the largest cohesive data

Table 4.2: Example data set for the multi-label encoding. c_1 correctly triggers rule 1, rule 2 and 179 are not triggered. c_2 incorrectly triggers rule 1, rule 2 is triggered corre-cltly, rule 179 is not triggered. c_3 incorrectly triggers rule 1, rule 2 is not triggered, and rule 179 is correctly triggered. c_{718} does not trigger rule 1, rule 2 and 179 are incorrectly triggered. Note that in this encoding, training and test use an identical schema. The only difference is that the prediction does not produce missing values.

	s_1	...	s_{547}	x_1	x_2	...	x_{179}	λ'_1	λ'_2	...	λ'_{179}	λ_1	λ_2	...	λ_{179}
c_1	+1	...	0	1	0	...	0	1	0	...	0	1	?	...	?
c_2	+1	...	+1	1	1	...	0	0	1	...	0	0	1	...	?
c_3	0	...	+1	1	0	...	1	0	0	...	1	0	?	...	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
c_{718}	0	...	0	0	1	...	1	0	0	...	0	?	0	...	0

25 Those 25 pesticides were also tested in our previous experiments investigating the sensitivity and selectivity of the method (see Table 6 in [28]). In addition, 22 other xenobiotics (pharmaceuticals)

set available because these compounds are made to be put into the environment and are among the most heavily regulated class of chemicals.

The matrix encoding described above was applied to the UM-BBD/UM-PPS from July 24, 2007, containing 1,084 compounds and 204 biotransformation rules (btrules). Of these, 366 compounds did not trigger by any rule (terminal compounds of reported pathways, compounds containing metals or other compounds whose biodegradation should not be predicted) were removed. Likewise, 25 strictly anaerobic (unlikely or very unlikely) btrules and btrules not triggered by any compounds in the UM-BBD were removed. Finally, 48 transformation rules triggered by only one structure were removed from the set as well. The remaining 718 UM-BBD compounds were submitted to 131 UM-PPS btrules. The predicate *triggered* was then defined to be true if a rule applied to a compound, and *observed* was defined to be true if the product could be found in the database.

The class distribution in the data set is very diverse (see Figure 4.3). There are only few transformation rules with both a balanced class distribution and a sufficient number of structures triggering them. Thus, we decided to implement classifiers for a subset of the transformation rules. We chose rules that provide at least a certain amount of information for the construction of the classifiers. The transformation rules needed to be triggered by at least 35 structures. On the other hand, for the ratio of “correct triggers”, we set a minimum of at least 0.15. These parameters seem sufficient to cover a sufficient number of cases and exclude overly skewed class distributions. Varying the parameters in further experiments did not lead to an improvement of the results.

Of the 131 transformation rules in the training set, this leaves 13 rules for the validation process (see Table 4.3). Considering the class distribution and number of examples of the remaining rules, it is not reasonable to learn classifiers for these transformation rules. However, to compare the results with previous work and to evaluate all transformation rules, we generated a *default classifier (DC)* for these rules which predicts the ratio of positive examples as the probability to produce a correct product. Thus, if the chosen threshold is below the ratio of positive examples, all structures are predicted as positive, i.e., they are predicted to trigger this transformation rule correctly.

For the computation of frequently occurring molecular fragments, we applied the FreeTreeMiner system [81], as it builds on a cheminformatics library to handle structures and substructures conveniently.

were only used for determining the reduction of predictions (see Table 4), because their degradation products are not known.

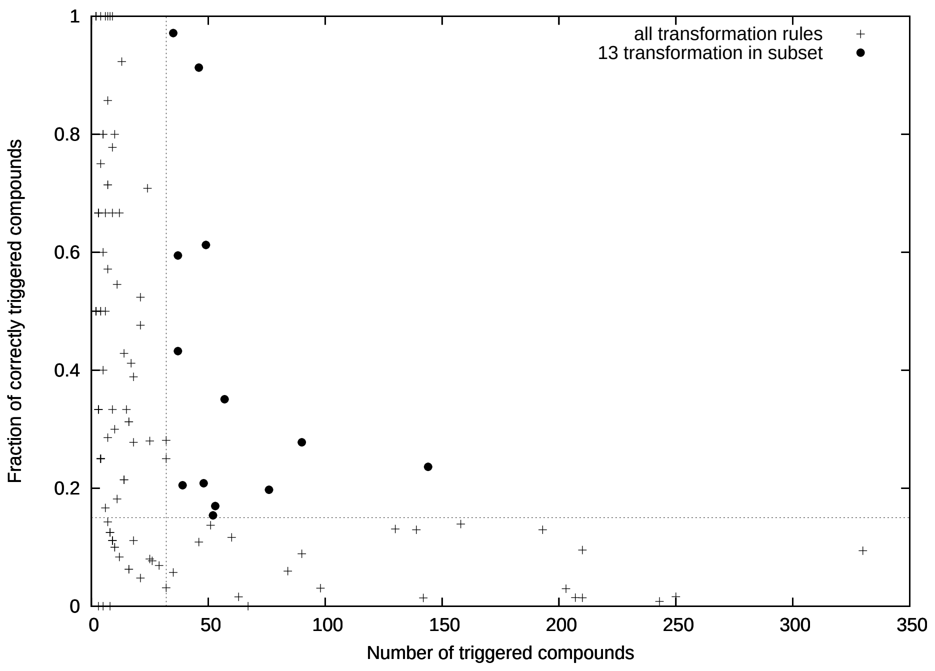


Figure 4.3: Characteristics of data sets for rules: size (number of triggered compounds) and class distribution (fraction of correctly triggered compounds). The 13 transformation rules used in the subset are marked. The dotted lines are the cutoffs (Number = 35, Fraction = 0.15) used to select the subset.

Table 4.3: List of the 13 transformation rules in the subset used for prediction.

Rule	Description
bt0001	primary Alcohol → Aldehyde
bt0002	secondary Alcohol → Ketone
	secondary Alcohol → Ester
bt0003	Aldehyde → Carboxylate
bt0008	vic-Dihydroxybenzenoid → extradiol ring cleavage
bt0029	organoHalide → RH
bt0036	aromatic Methyl → primary Alcohol
bt0040	1-Aldo/keto-2,4-diene-5-ol → Carboxylate + 1-ene-4-one
bt0055	1-carboxy-2-unsubstituted Aromatic → Catechol derivative
bt0060	vic-Hydroxycarboxyaromatic → Catechol derivative
	vic-Aminocarboxyaromatic → Catechol derivative
bt0063	primary Amine → Aldehyde or Ketone
	secondary Amine → Amine + Aldehyde or Ketone
	tertiary Amine → secondary Amine + Aldehyde or Ketone
	Methylammonium derivative → Trimethylamine + Aldehyde or Ketone
bt0065	1-Amino-2-unsubstituted aromatic → vic-Dihydroxyaromatic + Amine
bt0254	vic-Dihydroxyaromatic → intradiol ring cleavage
	vic-Dihydroxypyridine → intradiol ring cleavage
bt0255	vic-Dihydrodihydroxyaromatic → vic-Dihydroxyaromatic

4.4 Performance Measures

Clearly, we are facing a fundamental trade-off also found in many other applications of machine learning and classification: If the rules are too general, we will not miss many positive examples, but we might also include too many false positives. Vice versa, if the rules are too specific, we probably have few false positives, but we will potentially miss too many positives. It is convenient to think of this trade-off in terms of recall (sensitivity) and precision (selectivity). If the overall system predicts an observed product for a given substrate, we can count this as a true positive. If the system predicts a product that is not observed, we have a false positive. If a product is missing for a substrate, we have a false negative.[26] The number of true positives is denoted by TP, the number of false positives by FP, and the number of false negatives by FN. Then the standard definitions of recall (sensitivity) and precision (selectivity) can be applied:

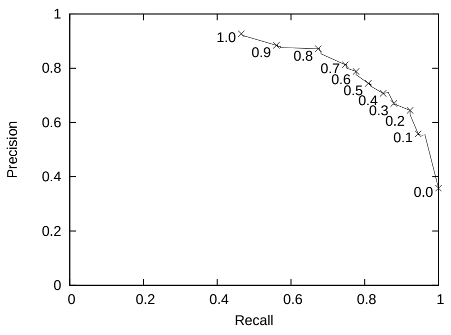
$$Recall = Sensitivity = \frac{TP}{TP + FN} \tag{4.8}$$

$$Precision = Selectivity = \frac{TP}{TP + FP} \tag{4.9}$$

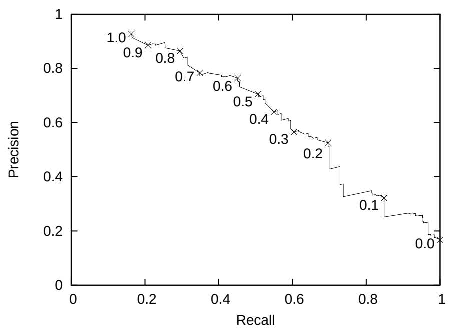
The overall number of products predicted by the system critically depends on the cut-off parameter θ. To evaluate the performance of the system, this parameter does not need to be fixed in advance. Instead, the parameter can be varied over the whole range from 0 to 1, and the resulting values for recall and precision can be plotted in two dimensions: recall is plotted on the x-axis, and precision on the y-axis. precision-recall plots (see Figure 4.4 and Figure 4.5) offer an easy and intuitive visualization of the trade-offs involved in choosing a certain value of θ. Also the results of approaches without cut-off parameters (e.g., relative reasoning as discussed above) appear as single data points in precision-recall space.

precision-recall analysis can be performed on the system level as well as on the level of individual rules. In principle, one could set the threshold individually for each rule, but this would introduce a large number of parameters. For simplicity, we chose to visualize the system's performance below by applying the same threshold for all rules. Also, as

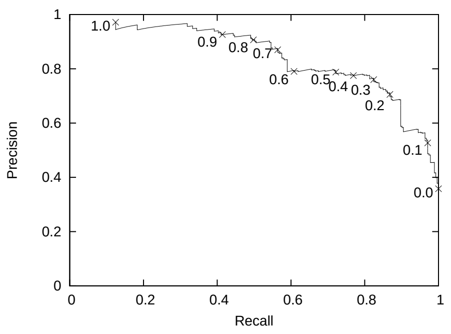
26 We count the false negatives in a slightly different way than in a previous paper [28], as we only consider products that are suggested by any of the biotransformation rules. In other words, we do not take into account products of reactions that are not subsumed by any of the rules. This is done because only for the products suggested by the UM-PPS, the method proposed here becomes effective – the classifiers can only restrict the rules, not extend them.



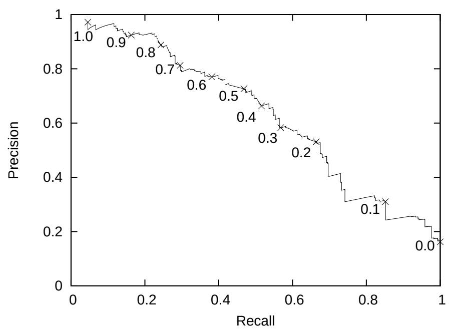
(a) Random forests on 13 rules.



(b) Random forest on 13 rules, default classifier on remaining rules.

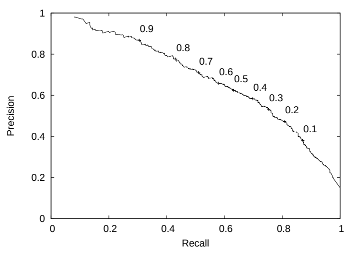


(c) SVM on 13 rules.

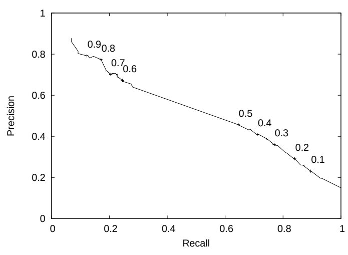


(d) SVM on 13 rules, default classifier on remaining rules.

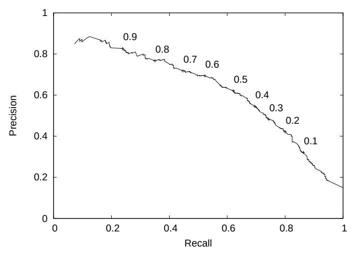
Figure 4.4: precision-recall plots from a *leave-one-out cross-validation* using the random forest classifier ((a) and (b)) and Support Vector Machines (c and d). On the left hand side (a and c), only the results of classifiers on a subset of the rules is shown. On the right hand side (b and d), classifiers were generated for the same subset and a default classifier is used for the remaining rules. The subset was chosen by using transformation rules with at least 35 triggered examples and a minimum ratio of known products of 0.15. Using these parameters, 13 transformation rules were selected. The threshold θ is given in ten steps per plot. Note that the points in precision-recall space are connected by lines just to highlight their position: In contrast to ROC space, it is not possible to interpolate linearly.



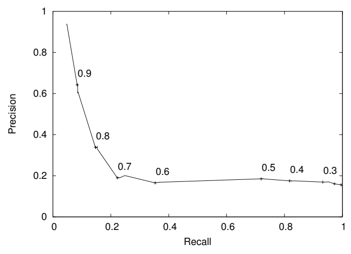
(a) Ensembles of Classifier Chains



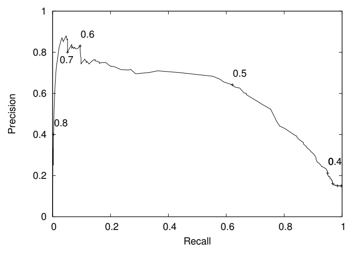
(b) Multi-Label Classification Using Boolean Matrix Decomposition



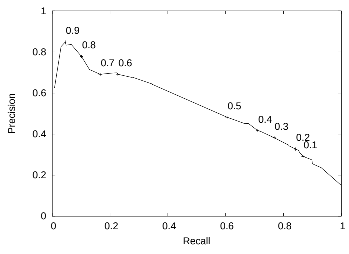
(c) Include Labels



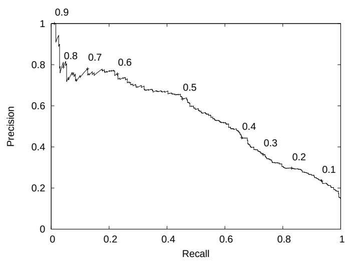
(d) Label Powerset



(e) Calibrated Label Ranking



(f) Binary Relevance2



(g) MLkNN

Figure 4.5: Precision-recall plots from a *leave-one-out cross-validation* using different multi-label classifiers on all rules as labels. The threshold θ is given in ten steps per plot, except in plots (d) and (e), where the lower thresholds values are too close to each other to differentiate among them in the plots. Note that the points in precision-recall space are connected by lines just to highlight their position: In contrast to ROC space, it is not possible to interpolate linearly.

individual classifier schemes should be sensitive and adaptive to different class distributions, one parameter for all should work reasonably well in the first approximation.

In addition to precision-recall analysis, we measure the area under the receiver operating characteristic (ROC) curve, which indicates the capability of a classifier to rank the examples correctly [13].

4.5 Experimental Results

In the following, we present the experimental results obtained with the proposed approach. After introducing various learning schemes and settings, we present the results on the xenobiotics test set and, more importantly, our main results from a *leave-one-out cross-validation* over the UM-BBD structures.

For the 13 transformation rules in the subset, we applied the random forest algorithm [9] in the implementation of the Weka workbench [37], because it gave good probability estimates in preliminary experiments. As a second classifier, we used Support Vector Machines trained using Sequential Minimal Optimization [73] in the implementation of the Weka workbench. We automatically adjusted the complexity constant of the Support Vector Machine for each transformation rule separately. We used a *10-fold cross-validation* to generate the data for the logistic models [73] to obtain well-calibrated class probability estimates.[27] As a default classifier on the remaining 118 transformation rules, we used the ZeroR algorithm of the Weka workbench.

In total, we evaluated four variants:

(a) 13 learned classifiers (*LC*) (i.e., random forests or SVMs) only,

(b) 13 learned classifiers (*LC*) and 118 default classifiers (*DC*),

(c) 131 learned classifiers (*LC*) (without default classifiers), and

(d) 1 learned multi-label classifier on 131 labels, equivalent to 131 learned single label classifiers (*LC*).

The idea of (a) is to evaluate the performance of the machine learning component of the system only. In (b), the overall performance of the system is evaluated, where 13 classifiers are complemented by 118 default classifiers. The purpose of (c) is to show

27 Note that any other machine learning algorithm for classification and, similarly, any other method for the computation of substructural or other molecular descriptors could be applied to the problem.

whether the default classifiers are really sufficient, or whether learned classifiers should be used even when samples are very small and classes are unequally distributed. (d) gives the results of extended multi-label evaluation on the data set. Multi-label classifiers can increase the total performance by exploiting label dependencies. The goal is to use all the rules for learning and gain the maximum performance. All the results are shown in terms of manually chosen points in precision-recall space (e.g., before inflection points) as well as the area under the ROC curve (AUC). The possibility to choose thresholds manually is one of the advantages of working in precision-recall and ROC space: Instead of fixing the precise thresholds in advance, it is possible to inspect the behavior over a whole range of cost settings, and set the threshold accordingly. Finally, we compare the results to the performance of relative reasoning.[28]

For compatibility with a previous paper [28], we start with the results of training on all UM-BBD compounds and testing on the set of 25 xenobiotics. The results are given in the upper part of Table 4.4. Note that in this case the default classifiers were “trained” on the class distributions of the UM-BBD training data and subsequently applied to the external xenobiotics test set.

First, we observe that ROC scores are on a fairly good level. The results in precision-recall space indicate that variant (b) is as good as variant (c). However, with an AUC of around 0.5, having 13 learned classifiers only variant (a) performs on the level of random guessing. An in-depth comparison of the two sets of structures (UM-BBD and xenobiotics) shows that this can be attributed to (i) the low structural similarity between the two sets, and (ii) the fact that a very limited set of rules trigger for the xenobiotics (a consequence of (i)). The average number of free tree substructures per compound is 48.76 in the xenobiotics data set, whereas it is 65.24 in the UM-BBD data set. Due to this structural dissimilarity, the transfer from one data set to the other is a difficult task. Therefore, we decided to perform a *leave-one-out cross-validation* over all UM-BBD compounds, where the structural similarity between test and training structures is higher than in the validation with the 25 xenobiotics as test structures.

Our main results from leave-one-out over the UM-BBD compounds are visualized in the precision-recall plots of Figure 4.4 and shown quantitatively in Table 4.4. In Figure 4.4, the top row (a and b) shows the plots of the random forest classifiers, the bottom row (c and d) displays the results of the Support Vector Machines. The left hand side (a

28 We cannot compare our results with those of CATABOL, because the system is proprietary and cannot be trained to predict the probability of individual rules – the pathway structure has to be fixed for training (for details we refer to Section 4.6). This means that CATABOL addresses a different problem than the approach presented here.

Table 4.4: Recall and precision for one threshold (on the predicted probability of being in the positive class) of the machine learning approach and for relative reasoning. The columns LC and DC indicate the number of transformation rules used for the learned classifiers (LC), SVMs or random forests, and the default classifiers (DC), ZeroR. The value of the threshold θ is determined manually considering the trade-off between recall and precision. We chose the threshold manually at an approximate optimum for recall and precision to provide a comparison to previous work [28]. Area under ROC Curve (AUC) is threshold-independent and only given for the new approach. The column with the variant refers to the assignment of rules to the different classifiers and is explained in the text. The multi-label classifiers are given with their used base classifier, in the case one was required. *Ensembles of Classifier Chains* (ECC) (see Section 2.1.4.3), *Multi-Label Classification using Boolean Matrix Decomposition* (MLC-BMaD) (see Section 3.1.2), *Include Labels* (IL) [95], *Label Powerset* (LP) (see Section 2.1.4.6), *Binary Relevance2* (BR2) (see Section 2.1.4.2), and Calibrated Label Ranking (CLR) (see Section 2.1.4.5) all use random forests (RF) as base classifiers. *Multi-Label k Nearest Neighbor* (MLkNN) (see Section 2.1.5.1) does not require any base classifier.

	Method	Variant	LC	DC	θ	Recall	Precision	AUC
Xeno-	RF	(a)	13	0	0.417	0.400	0.333	0.505
biotics	**RF**	**(b)**	**13**	**118**	**0.296**	**0.525**	**0.447**	**0.676**
	RF	(c)	131	0	0.35	0.475	0.404	0.664
	SVM	(a)	13	0	0.023	0.800	0.235	0.389
	SVM	**(b)**	**13**	**118**	**0.296**	**0.475**	**0.463**	**0.674**
	SVM	(c)	131	0	0.157	0.410	0.390	0.599
	RR	-	-	-	-	0.950	0.242	-
UM-	RF	(a)	13	0	0.600	0.777	0.788	0.902
BBD	**RF**	**(b)**	**13**	**118**	**0.308**	**0.595**	**0.594**	**0.842**
	RF	(c)	131	0	0.485	0.653	0.632	0.857
	SVM	(a)	13	0	0.329	0.813	0.771	0.903
	SVM	**(b)**	**13**	**118**	**0.294**	**0.582**	**0.588**	**0.841**
	SVM	(c)	131	0	0.250	0.632	0.623	0.833
	ECC/RF	**(d)**	**131**	**0**	**0.510**	**0.630**	**0.628**	**0.894**
	MLC-BMaD/RF	**(d)**	**131**	**0**	**0.500**	**0.615**	**0.457**	**0.821**
	IL/RF	(d)	131	0	0.499	0.623	0.620	0.863
	MLkNN	(d)	131	0	0.477	0.557	0.553	0.844
	LP/RF	(d)	131	0	0.700	0.223	0.190	0.597
	BR2/RF	(d)	131	0	0.500	0.605	0.482	0.841
	CLR	**(d)**	**131**	**0**	**0.497**	**0.629**	**0.631**	**0.873**
	RR	-	-	-	-	0.942	0.267	-

and c) shows the results of the classifiers on the 13 transformation rules in the subset, the right hand side (b and d) uses the default classifier for the remaining rules. The plots tend to flatten when including predictions of the default classifier.

The overall performance does not differ too much between random forest and Support Vector Machines. Using both classification methods, we can achieve recall and precision of slightly less than 0.8 (see Figure 4.4 and also the values for variant (a) in the lower part of Table 4.4) for the *LC* only in a *leave-one-out cross-validation.* The quantitative results in Table 4.4 also show that the performance of random forests and support vector machines are on a similar level. Also, in this case, the performance of learned classifiers complemented by default classifiers is nearly as good as the performance of the learned classifiers for all rules, supporting the idea of having such a mixed (*LC* and *DC*) approach.[29] However, as expected, the machine learning component consisting of 13 learned classifiers only performs better on average than the overall system with 118 default classifiers added in this case. In summary, the AUC scores are satisfactory, and the precision-recall scores of approximately 0.8 of the machine learning component show that improvements in precision are possible without compromising recall too much. Therefore, the machine learning approach provides some added value compared to the relative reasoning approach developed previously [28].

To evaluate the enhancement of using both the structural information and the expert knowledge in the transformation rules, we applied the new method individually to the data set leaving out the structure and, in a second run, the transformation rules. As it tends to give smoother probability estimates, here and in the remainder of the section we focus on random forest classifiers. Using only the structural information gives an AUC of 0.895, whereas the transformation rules only give an AUC of 0.885. Taken together, we can observe an AUC of 0.902, which, despite the apparent redundancy for the given data set, marks an improvement over the results of the individual feature sets.

In the multi-label experiments, we could improve the overall performance. We used 7 multi-label classifiers to predict the probability of the biodegradation rules. Due to the higher complexity of the algorithms, the experiments are limited to random forests as base classifiers where needed. SVMs increase the running time of multi-label classifiers drastically. *Ensemble of Classifier Chains* (ECC) by Read *et al.* [77] (see Section 2.1.4.3) showed the most promising results. Using all 131 rules in the prediction, the algorithm almost gained the same performance level regarding the area under the ROC curve as single-label SVMs on only the 13 selected rules. This provides the possibility to include

29 In other words, it shows that informed classifiers do not pay off for the rest of the rules.

a greater number of rules in the classification with some benefit. Single-label classifiers do not give better than random performance on the additional 118 rules.

The other multi-label classifiers all achieve a good performance with only the exception of *Label Powerset*. This algorithm does not perform that well on data sets with a large number of unique label combinations. Given the size of the data set and the rather large number of labels, this is the case for this data set. This explains the almost random performance of this classifier. The advantages of multi-label classifiers can be seen regarding recall and precision. *ECC* raises both values compared to the single-label SVM case while at the same time using more labels in the prediction. None of them is as biased towards one of the measures as Relative Reasoning. The precision-recall plots (see Figure 4.5) give smoother plots than the single-label predictions. In all but one case one can see and choose a threshold for addressing the trade-off between the measures. The LP method again shows that it cannot perform well on this data set.

An example prediction of the biotransformation of a structure is given in Figure 4.6. We applied our approach to Amitraz, a pesticide from the xenobiotics data set. The incorrectly triggered transformation rules all obtain a rather low probability, whereas bt0063, a correctly triggered rule, is the only transformation rule being predicted with a probability higher than 0.53. As the xenobiotics data set is very small, we generated random forest classifiers for every transformation rule triggered by this structure for the purpose of the example.

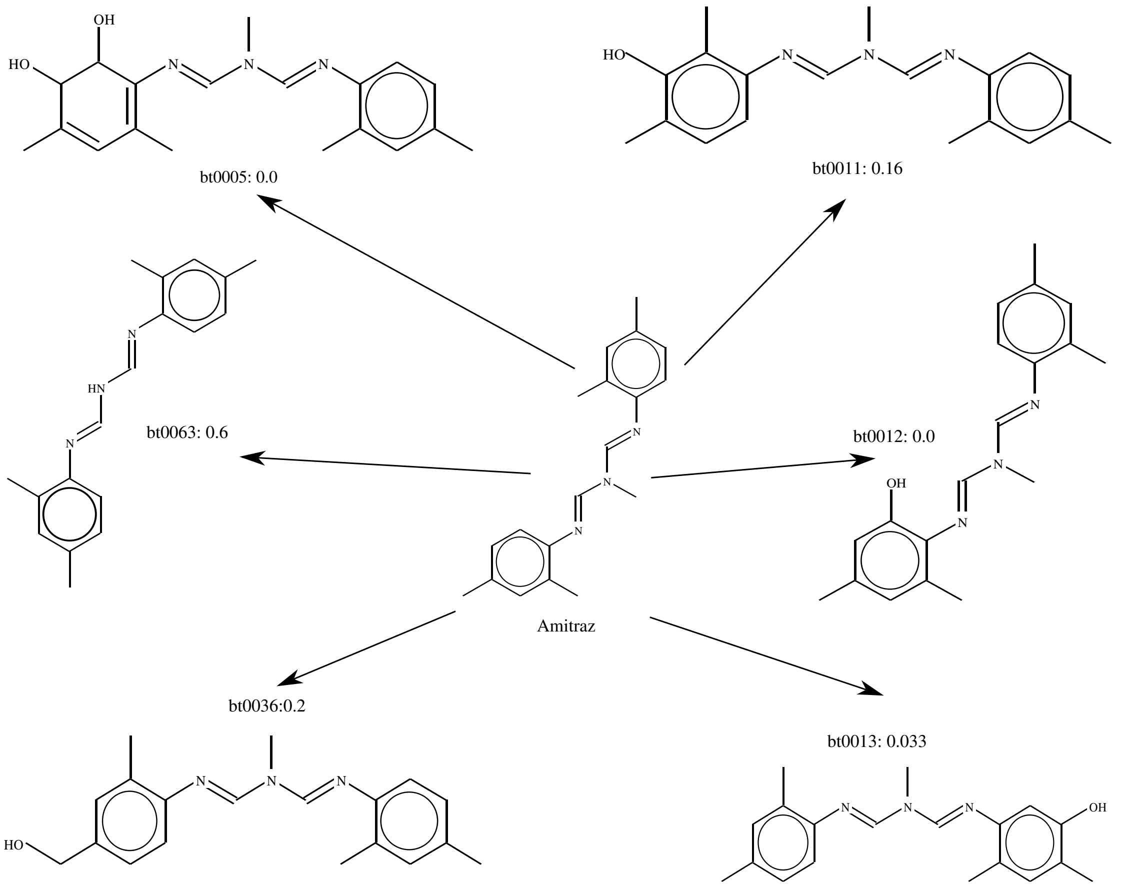


Figure 4.6: Application of the new method to Amitraz, a compound from the xenobiotics data set. For each transformation rule triggered by this structure, an example product is given. Some of the transformation rules can produce more than one product from this structure. We applied random forest classifiers to the structure. The numbers indicate the predicted probability that the corresponding transformation rule produces a known product. From the transformations predicted by the UM-PPS, only bt0063 produces a known product. As shown in the figure, this is the only transformation rule with a relatively high predicted probability.

4.6 Discussion and Conclusion

We presented a combined knowledge-based and machine learning-based approach to the prediction of biodegradation products and pathways, which performs relative reasoning in a machine learning framework. One of the advantages of the approach is that probability estimates are obtained for each biotransformation rule. Thus, the results are tunable and can be analyzed in precision-recall space. Making the trade-off between recall (sensitivity) and precision (selectivity) explicit, one can choose whether one or the other is more important.

In contrast to CATABOL, the approach works on the level of rules and not on the level of pathways. In CATABOL, the structure of pathways has to be laid out in advance in order to solve the equations based on the training data. To make the computations more stable, reactions have to be grouped using expert knowledge. In contrast, we apply the rules to the training structures to extract a matrix, which is the basis for the creation of the training sets for each rule. Multi-label classifiers showed that they can improve the overall performance of the models and include more rules in the prediction without losing performance when predicting rules for new structures. CATABOL learns parameters for a fixed pathway structure, whereas the approach proposed here learns classifiers for (individual) transformation rules. During testing, only the pathways laid out for training can be used for making predictions in CATABOL. In contrast, the approach presented here predicts one transformation after the other according to the rules' applicability and priority determined by the classifiers. Overall, the training of CATABOL requires more human intervention than our approach, e.g., for grouping and defining hierarchies of rules (see Dimitrov *et al.* [18]).

One might speculate (i) which other methods could be used to address this problem, and (ii) where the proposed solution could be applied elsewhere. Regarding (i), it appears unlikely that human domain experts would be able and willing to write complex relative reasoning rules as the ones derived in this work. Alternatively, other machine learning schemes could be used to solve the problem, for instance, methods for the prediction of structured output [49]. These methods should be expected to require a large number of observations to make meaningful predictions. Also, with the availability of transformation rules, the output space is already structured and apparently much easier to handle than the typically less constrained problem of structured output. Regarding (ii), the approach could be used wherever expert-provided over-general transformation rules need to be restricted and knowledge about transformation products is available. It would be tempting to use the same kind of approach for other pathway databases like

KEGG, if they were extended towards pathway prediction systems like the UM-BBD. Our extended pathway prediction system could also be used as a tool in combination with toxicity prediction, as the toxicity of transformation products often exceeds the toxicity of their parent compounds [86]. The procedure would be first to predict the degradation products and then use some (Q)SAR model to predict their toxicity.

Currently, we are integrating the resulting approach into an experimental version of the UM-PPS server. In the future, it may become necessary to adapt the method to more complex rule sets, e.g., (super-)rules composed of other (sub-)rules. Such complex rule sets should be useful for the representation of cascades of reactions.

CHAPTER 5

Predictive Toxicology

As discussed in the context of biodegradation pathway prediction in Chapter 4, *in silico* methods to predict toxic effects of chemicals are increasingly sought. Currently, it is required to carry out a vast number of animal experiments to determine the toxicity of a new chemical. There are many reasons to minimize that number. Above all, animal experiments are ethically disputed. Secondly, they tend to be expensive and difficult to execute. Nevertheless, recently published regulations on chemical substances by the European Union (REACH) require a large number of new tests, which would be expensive in terms of money and lives of animals. Hence, this causes a need for the computer-aided prediction of toxic effects (*in silico* prediction) to reduce the cost both of money and animal lives.

Sophisticated computer models could simulate the *in vivo* experiments and predict the outcome *in silico*. The area of predictive toxicology aims to learn these *in silico* models to predict toxic effects. Predictive toxicology made significant progress in recent years, with OpenTox [39] being one of the projects. It aimed to generate and eventually implemented on-line web services to provide an interface to predictive models and frequently used tools in predictive toxicology. Another project launched in the past few years is the ToxCast™ project by the EPA. The aim of this project is to break the *in vivo*/*in vitro* border, i.e. predict toxic effects using data from *in vitro* experiments. To achieve this, the project provides extensive *in vivo* data, generated in numerous animal experiments, together with a comprehensive collection of *in vitro* data generated using biochemical assays. This data is provided for a set of approximately 300 environmental chemicals. However, the long-term plans of the EPA is to provide data for a larger number of chemicals.

The overall goal of the project is to learn predictive models for the *in vivo* endpoints[30]. At the current state, the project organizers try to find a way to prioritize further data generation and evaluate the current data set. Nevertheless, previous work showed that the quality of the data set for learning predictive models is disputed. While there exist some publications that claim it is possible to learn meaningful models, others claim it is not possible. However, previous work focused on learning models on single *in vivo* endpoints using the *in vitro* data, which leaves a vast amount of information present in the *in vivo* data. Clearly, there exist dependencies between the individual endpoints. In the following, we propose to use the dependencies in the *in vivo* data to improve the performance of the predictive models. The structure of the data set makes it predestined for multi-label methods, which allow to exploit the dependencies.

The remaining chapter is organized as follows. First, we give an introduction to the data set and previous work. This is followed by a detailed overview of the chemicals, *in vitro* and *in vivo* data. Subsequently, we analyze the multi-label properties of the data set and we explain our approach to learn models on ToxCast™. Subsequently, we give the results of an experimental evaluation. Finally, we present a conclusion and an outlook to future work.

5.1 The ToxCast™ Data Set

The first version of the ToxCast™ data set (phase I) was released by the EPA in 2009. The goal of the project is to develop models that use *in vitro* assays, physico-chemical properties and chemical descriptors to predict toxicological endpoints [22, 55, 59, 21, 64, 66]. The model training can be based on a large database of *in vivo* endpoints, which were generated over several decades. Overall, this project aims to break the *in vivo* / *in vitro* border: If well-performing models can be learned using the ToxCast™ data set, the results of animal experiments can be predicted by *in silico* models using only the outcome of an *in vitro* experiment as input data.

The ToxCast™ project has the potential of generating large amounts of data to learn high quality models predicting in vivo endpoints. Nevertheless, at the moment there is still only the phase I data set published. The EPA plans to publish phase II in the near future and phase III thereafter. The upcoming phases will dramatically increase the size of the data set. The main increase will be the number of tested chemicals. This

30 In this case, the endpoints are simply the single columns, i.e., toxic effects. In predictive toxicology, the term "endpoint" refers to the target outcome of an experiment

will enable to learn more meaningful models on the chemicals for a broader range of chemicals. So far, the chemicals represent a tiny fraction of the chemical space (only 309 substances are tested so far) [21, 55, 22]. Despite this small number, the data set can already be used to evaluate algorithms and methods generating models on this type of data. Evaluation of the models can help to prioritize chemicals and endpoints for further testing, i.e., models can identify endpoints with a high potential to provide useful data for the models. Future experiments can focus on these high-quality endpoints.

The first work on the ToxCast™ data set was published and discussed at the ToxCast™ Data Analysis Summit in May 2009[31]. Among others, Barry Hardy presented preliminary work of the OpenTox project on ToxCast™. This work focused on two approaches: First, on applying existing toxicity models to the chemicals, and comparing the results with the *in vivo* endpoints. The second approach was to learn a model on a single *in vivo* endpoint, thereby evaluating the performance. None of the proposed approaches showed promising results. It was suggested that either there are errors in the data or the tested structures are too different from the structures used to learn the models.

Another approach was presented by Alexander Tropsha. He proposed to first build a model that predicts whether a structure is predictable by a final model to predict toxicity. The final model predicts if the structure is toxic in terms of *in vivo* data as well as *in vitro* data, i.e., both data agree on the toxicity. Using only the structure agreeing on these terms, consensus models for liver toxicity were built. Thus, instances are not taken into account that could be used in the classification. This approach is not easy to justify from a machine learning point of view. Even if models trained on *in vitro* data state the opposite of models trained on *in vivo* data, it might be used by the classifiers, but meaningful predictions will not be given for a structure as they are sorted out by this approach. Still, this approach sorts out structures that might confuse the model, and, on the prediction level, structures that are hard to predict.

Despite this preliminary work, the total number of publications using the ToxCast™ data set to learn models for predicting *in vivo* endpoints is quite low. Moreover, most publications are provided by the EPA group working on the project. In general, publications using the ToxCast™ data set indicate a mixed opinion on the quality of the data. Nevertheless, Thomas *et al.* [92] recently published a study of the data set. They used

31 While the results of the preliminary work was never officially published, the presentations are still available at the summit's web page `http://www.epa.gov/ncct/toxcast/summit.html`. Despite the fact that some of the approaches seemed promising, they were hardly carried on. Hence, most of the work presented at this summit was never published.

the *in vitro* data to build multiple models predicting the *in vivo* data. The models were evaluated using a *cross-validation* and model performance was investigated on different endpoints. The authors state that they observed a performance which is not as good as results published in previous publications. They suggest the reason for the bad results might be that the *in vitro* data does not reflect the biological processes involved in the *in vivo* endpoints.

Dix *et al.* [21] replied in a letter to the editor to Thomas *et al.* [92] that the latter approach did not succeed as it did not use biological knowledge about the data set. They state that the EPA did obtain good results in previous experiments using feature selection and background knowledge. Dix *et al.* believed that the low quality of the predictive models by Thomas *et al.* resulted from the used methodology and is not due to a systematic problem with the ToxCast™ data set.

Martin *et al.* [65] focused on a subset of the *in vivo* data, describing reproductive toxicity. They used univariate feature selection based on correlation coefficients, chi-square tests, and t-tests. The assays were grouped by corresponding gene or assay family and assigned to *in vivo* endpoints. In addition, chemicals with little or no *in vitro* activity on these endpoints were excluded. They argued that chemicals with a lack of an *in vitro* activity cannot be used for models predicting *in vivo* endpoints, independent of their true toxic effects. Based on the subsets of features, models were generated using linear discriminant analysis (LDA). To evaluate the models, a five fold *cross-validation* was performed. In addition, a subset of the chemicals was used to perform a hold out evaluation. The authors achieved an overall balanced accuracy of above 80% on the used *in vivo* endpoints when using hold out validation. Using *cross-validation*, the balanced accuracy was at 74%.

Similarly, following the main goal to find correlations between *in vitro* and *in vivo* endpoints in ToxCast™, Sipes *et al.* [87] used again LDA to generate predictive models. In a first step, a feature selection was performed using univariate statistics, i.e., Pearson's correlation coefficient, students t-test and the chi-squared statistic. In the next step, they combined significant assays by genes into sets. Not significant assays were condensed by group (i.e., gene family, pathway, etc.). Finally, single assays, not combined with others, were combined into a separate set. On these sets, LDA was performed, and the whole process was evaluated by a *cross-validation*. In both rat and rabbit, the results showed a sensitivity and specificity above 70% and a balanced accuracy of 70%.

A similar approach was used by Kleinstreuer *et al.* [56]. Their goal was to identify signatures for potential chemical disruption of blood vessel formation and remodeling. The ToxCast™ chemicals were ranked according to the vascular bio-activity score. Subsequently, LDA was performed on the corresponding *in vivo* data. The authors were able to achieve an AUC of 0.86 on rabbit and 0.9 on rat. Balanced accuracy using *cross-validation* was at 80% (rabbit) and 90% (rat).

So far, the data set was used to test existing models against the new endpoints, to generate new models using the new *in vitro* data, or to examine relations between *in vitro* and *in vivo* data regarding certain toxicity. Most approaches focused on building models for one specific endpoint in the data set or summarized a group of endpoints. Given the large number of target values and the high number of co-occurrences of these target values (on average 24.11 endpoints per chemical), this data set seems to be very suitable for multi-label classification. Most of the target values are related to each other, they concern organs either in mouse, rat or rabbit, some address toxic effects in the same organ. This naturally leads to large number of interdependencies among the target values or labels, thus the performance of classifiers trained on the training data should be expected to improve by using multi-label approaches.

The only approach using multi-label classifiers was reported by Jeliazkova *et al.* presented at the above mentioned ToxCast™ data analysis summit, and is hence not officially published. As a multi-label classifier, predictive clustering trees [7] were used. In their preliminary experiments, the multi-label classifiers clearly improved the performance over using similar single-label classifiers.

Currently, the data consists of 309 chemical structures, mostly pesticides. For each of these chemicals, multiple *in vivo* experiments were carried out [51, 64]. In total, there are 463 *in vivo* target values in the data set. The descriptors and *in vitro* endpoints sum up to 1697 features of the chemicals. The data set is publicly available via the ACTOR system [53].

5.1.1 Chemicals

In phase I and phase II, the EPA tested 1060 chemicals. At the moment, only 309 chemicals belonging to phase I are publicly available, as phase II is not yet finished. The published chemicals are all environmental chemicals, i.e., chemicals being released to the environment, e.g. pesticides. These chemicals can enter the human metabolism via the food chain and thus can cause toxic effects. The majority of the chemicals in ToxCast™ are registered active pesticides (273 of the 309 structures). Only 18 chemicals are de-registered active pesticides, 22 are inert pesticides, and 33 antimicrobials. These chemicals represent a general profile of chemicals that are currently used in agriculture. A detailed list of the used chemicals is given in Appendix A.1.

5.1.2 In Vitro Assays

The chemicals were tested on multiple *in vitro* assays. Most of these data were generated by biochemical companies. Each assay is related to certain biological functions. It should be noted that the results of the assays are not available for all ToxCast™ substances. Almost every assay has gaps in the data. These missing values are sometimes attributed to problems in the experiments, i.e., to failed experiments. More often, a whole block of chemicals was not tested in certain assays or the compound could not be used in the experiments. In the following, I will give a short overview of each *in vitro* assay.

5.1.2.1 ACEA

The ACEA assay [63] was provided by ACEA Biosiences Inc[32]. It is based on human lung carcinoma cell lines. It gives the toxic effect of the substances on these cell lines together with their mechanistic interpretation. The assay is grouped into seven features that capture the response of the cells to the substance. During the exposure, the response of the cell lines is given in real time. From this, the lowest effect levels (LEL) and IC50 values are calculated. More specifically, the IC50 gives the amount of the chemicals that is needed to reduce the process of the cell line by half. The LEL gives the lowest level of concentration for which an effect on the cell line can be observed.

5.1.2.2 Attagene

Attagene[33] provided 73 features describing the modulation of transcription factor activity in human hepatome HepG2 cells caused by the structures. The features were generated by two reporter gene assays, that is, by Cis-Factorial assays and Trans-Factorial assays. The Cis-Factorial assays, providing 52 features, describe the change in activity of the transcription factors. The Trans-Factorial assays, providing 29 features, consist of agonistic or antagonistic properties of chemical compounds across human hormone nuclear receptors. The effects of each substance were tested at seven concentrations, and values were measured in a 3-fold serial dilution over an exposure of 24 hours. Similar to the ACEA data set, the LEL and IC50 values are given in the features.

32 `http://www.aceabio.com`
33 `http://www.attagene.com`

5.1.2.3 Bioseek

Bioseek Inc.[34] provided 173 features modeling human tissue and disease states [5]. They give the effect of chemicals on signaling pathway structures. The effects were tested at 4 concentrations and measured over an exposure of 24 hours. On this data, the lowest effect levels are calculated. Effects were compared to a profile database storing biological responses to drugs. In the assays, up- and down-regulation is distinguished.

5.1.2.4 Cellumen

Cellumen[35] used automated fluorescent microscopy to generate 57 features [33]. The features describe the effects of the structures on toxicity biomarkers in human cell cultures of HepG2 and rat (Ratus Norvegicus) hepatocytes. Structures were tested at 10 concentrations. Effects were measured three times in 72 hours. From this data, the activity concentration 50% (AC50) was calculated. The AC50 gives the concentration of the structure where the activity in the cell line is reduced to 50%.

5.1.2.5 CellzDirect

The data set provided by CellzDirect[36] consists of 42 features regarding exclusively human cell lines describing 16 genes [80]. The cell lines were exposed to the chemicals at 6, 24, and 48 hours. A new batch of chemicals was added to the cell lines every day. In the experiments, five different concentrations were tested. The results of the experiments are normalized and lowest effect levels are calculated.

5.1.2.6 Gentronix

Only one feature is provided by the assays of Gentronix[37]. It describes the human GADD45a promoter activity in the transformed cell line. Three concentrations were tested over 24 and 48 hours. If the cytotoxic activity was recognized in the first step, the substances are retested at a lower concentration. From these experiments, the lowest effect level is calculated as well.

34 `http://www.bioseekinc.com`
35 `http://www.cellumen.com`
36 `http://www.cellzdirect.com`, acquired by Invitrogen, see `http://www.invitrogen.com`
37 `http://www.gentronix.co.uk`

5.1.2.7 NCGC

The NIH Chemical Genomics Center (NCGC)[38] contributed three features to the ToxCast™ data set [46, 48, 85]. The features represent the reactions of nuclear receptors in human and rat cell lines. Concentration response curves were generated from this data. The curves were used to calculate the AC50 values of the structures regarding these cell lines.

5.1.2.8 Solidus

The four Solidus[39] features describe response of Algionate-immobilized Phase I and Phase II enzymes to the structures. The assays were exposed for six hours at nine different concentrations. LC50 values are calculated for four different assay conditions (each resulting in one feature). In LC50, LC refers to lethal concentration, the value gives the concentration of a substance where 50% of a population or cell line dies. The assay consists of a control line, Phase I enzymes, Phase II enzymes, and both Phase I and Phase II enzymes.

5.1.2.9 Novascreen

The biggest part of the assays is the Novascreen data set [58]. The data set consists of 239 features. In contrast to the previous assays, which are cell-based, Novascreen is a biochemical assay. This means it does not describe the reaction of cell lines to the substances but the reactions of proteins, e.g., the change in enzyme activity or ligand bindings. Several protein super families are included (e.g., Kinase, Phosphatase, Protease). The proteins were tested against two concentrations and the data are normalized. The values give the percentage of activity change when the structure is added, compared to a control activity. The data itself represents multiple formats, e.g., radioligand receptor binding, fluorescent receptor binding, fluorescent enzyme substrate-intensity quench, or fluorescent enzyme substrate-mobility shift.

5.1.3 Chemical Descriptors

In the initial release of the ToxCast™ data set, the structures were only described by *in vitro* and *in vivo* assays. Later, additional *in silico* descriptors were added to the data set, calculated by various cheminformatics tools.

38 `http://www.ncats.nih.gov/research/reengineering/ncgc/ncgc.html`
39 `http://www.solidusbio.com`

5.1.3.1 ChemClass

This data set represents 103 chemical classes. Each chemical can belong to multiple chemical classes. The value in a feature is *true* if the structure belongs to the corresponding class, and *false* if it does not belong to that class. This data set was provided by the EPA as supplementary information to the chemicals. The class is included in this data set. As the class provides information on the function of the compounds, classifiers for toxic effects can be improved using this information.

5.1.3.2 EPISuite

The Estimation Programs Interface Suite (EPISuite) is a software developed by the EPA. It calculates certain physical, chemical descriptors, and estimations on the environmental fate. EPISuite was used to calculate these descriptors on the compounds, which resulted in 30 features describing the structures.

5.1.3.3 Molecular Networks MetaboGen

This data set represents a reactivity matrix. For each structure, possible reactions are given in the features. These features are calculated by software from Molecular Networks[40]. The possibility for 125 reactions are given per structure.

5.1.3.4 Molecular Networks Whole Molecule Properties

The second data set of Molecular Networks summarizes 15 properties of the molecules, such as the molecular weight, the number of atoms, or polarization.

5.1.3.5 PhysChem Derived

This data set provides only 2 features, the LogP value and LogP-TPSA. The LogP value is a measure of the hydrophility of a substance. It is the hydrophilicity logarithm of the ratio of a compound solved in octanol to the ratio of substance solved in un-ionized water. If the value is positive, the substance is hydrophobic, if the value is negative, the substance is hydrophilic.

40 `http://www.molecular-networks.com`

5.1.3.6 LeadScope

Leadscope[41] is a software company providing a commercial tool to calculate descriptors on chemical structures. For the ToxCast™ structures, 6 features were calculated and included in this data set. These features are rather simple descriptors like the LogP value or the number of rotatable bonds.

5.1.3.7 QikProp

This data set includes 49 descriptors calculated by the QikProp software by Schrödinger[42]. These descriptors are calculated on the 3D structure of the compounds. The data set provides features like number of amides, number of amines or number of ring atoms, and more sophisticated descriptors.

5.1.3.8 Structure Classifiers

This data set consists of 241 chemical classes indicating if the structure can be assigned to that class. Examples for the classes are alcohol alkenyl, ether spiro, or sulfone aromatic. This can be used in the model building, as it is a description of the structure.

5.1.4 In Vivo Endpoints - ToxRefDB

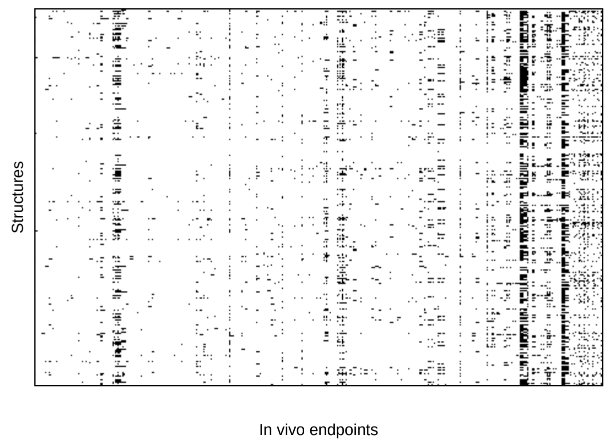
The target variables of the data set are, as explained above, the *in vivo* endpoints. The endpoints were measured in numerous animal experiments using mice, rabbits, and rats. The assays consist of rat or mouse 2-year cancer or chronic bioassays, rat multi-generational reproductive toxicity assays, and rat and rabbit prenatal developmental toxicity assays [52]. Each value in the data set is the lowest effective level that has been measured. This means the lowest amount (milligram) of structure per kilogram body weight per day an animal needs to be given to observe a statistically significant raise of a cancerous effect compared to a negative control. Note that not all substances were tested on all endpoints. Approximately 250 structures are given per endpoint.

Figure 5.1a shows a heatmap of the *in vivo* data set. The x-axis gives the *in vivo* endpoints, the y-axis shows the chemical structures. Each point is a positive value, i.e., the structure on the y-axis causes the toxic effect on the x-axis. If there is no point given, there is either no toxic effect, or this experiment was not successfully executed. The heat-map shows that there are certain endpoints that are more common (visible "columns" in

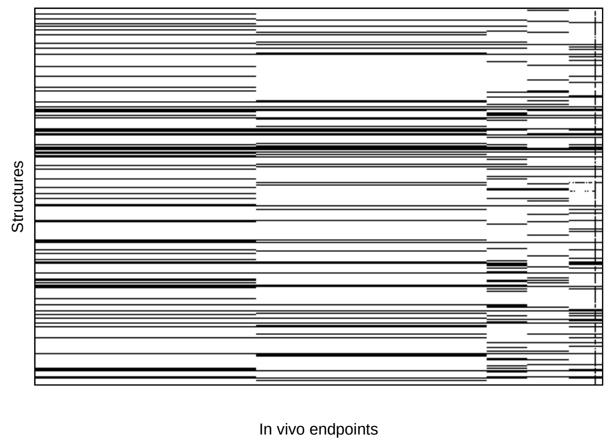
41 `http://www.leadscope.com`
42 `http://www.schrodinger.com`

the plot) than others. Additionally, one can clearly see that certain chemicals are rather toxic (“lines” in the plot), while others do not cause many toxic effects.

The missing values in the *in vivo* data are shown in Figure 5.1b. The x-axis gives the *in vivo* endpoints, the y-axis shows the chemical structures. A point in the heatmap represents a missing value. If there is no point, the experiment was successfully carried out, meaning that it is known if the toxic effect occurs. Certain blocks in the data are clearly visible. This is due to certain chemicals being systematically left out in certain experiments.



(a) Positive Labels



(b) Missing Values

Figure 5.1: Heatmap of the ToxCast™ data set. On the y-axis, the chemicals are plotted, the x-axis gives the *in vivo* endpoints. Note that the values of the heatmap are binary, hence only black and white as colors exist. Each black point in the heatmap indicates a positive label. In Figure (a), a positive label indicates that the structure on the y-axis causes the toxic effect on the x-axis. The chemicals and endpoints are not sorted, they are given in the exact order as provided in the data set. In Figure (b), each point in the heatmap indicates a missing value. If there is no point, it is known if the compound causes the corresponding toxic effect.

5.2 A View on ToxCast™ from the Multi-Label Perspective

As discussed above, related work gives some indications that the ToxCast™ data set might be hard to use for predictive toxicology. On the other hand, these publications focused mainly on applying already existing models on the data set or learning classifiers on a very limited number of *in vivo* endpoints (in most cases only one endpoint). Rarely, the data set is considered as a multi-label data set, even though using multi-label classification for ToxCast™ seems natural: the endpoints depend on each other; many of them describe similar toxic effects, regarding closely related animals (rat, rabbit and mouse), or even affecting the same organ. More precisely, if a toxic effect is caused by a compound in one animal, the same or a similar compound will most probably cause this effect in closely related animals. The same applies for organs as well. If one organ is affected by a compound in a way, it might affect the same organ in other ways too. Thus, if a certain endpoint or label is positive, there are certainly related labels which are positive too. These dependencies can be exploited by multi-label classifiers, and therefore multi-label classifiers are likely to improve the quality of the models. It has already been shown that it is possible to generate predictive models for certain endpoints[43].

Whereas the overall layout of the data set shows promising statistics for multi-label classification, there are some drawbacks to be considered. One of the challenges of the ToxCast™ data set is its sparseness. Given the large number of toxic endpoints, most compounds do not cause a large fraction of these toxic effects. Moreover, there exist many missing values, as compounds were not tested against all endpoints or certain experiments failed. Thus, there exists no information whether the toxic effect is caused by the substance.

On the other hand, general multi-label statistics seem promising (see Table 5.1). The data set has a relatively large number of labels with only a few instances. This is a good indicator for multi-label data sets. If there were more instances, additional label information would not be needed, as there would be enough data for each label to be predicted. With only few labels, the dependencies might not be well represented in the data. Thus, to use the benefits of multi-label classification, multi-label data sets should be large in terms of features and labels and rather small in terms of instances. Additionally, the large number of positive labels per instance (cardinality)[44] implies that

43 A summary and defense of the data set is given by Dix *et al.* [21].

44 The value of 24.11 for the cardinality might not seem that high. Nevertheless, compared to other multi-label data sets, one can see that this value is hardly reached by standard multi-label data sets (e.g., see Table 3.2).

Table 5.1: Summary statistics of the ToxCast™ data set. With respect to other multi-label data sets, ToxCast™ has a high number of labels compared to a low number of instances. The missing values are uncommon in multi-label data sets. There are, to this point, no publications addressing the problem of missing values in the labels. Cardinality is high compared to other data sets. This indicates that multi-label classifiers can perform well on ToxCast™, as it implies high dependencies between the labels. The density, on the other hand, is low, which is due to the high number of labels. This shows that an instance has only a small ratio of labels. Therefore, there might be not a big overlap on label patterns in the data set. This can be seen also by the high number of distinct label combinations. There is almost one unique combination per instance, which makes it hard for many multi-label classifiers to perform well on this data set (e.g. label power set).

Statistic	Value
number of labels	463
number of instances	309
number of features	1697
cardinality	24.11
density	0.05
distinct label combinations	308
percentage of missing values in labels	21.76

there are many dependencies in the labels. On the other hand, the low density can be attributed to the high number of labels. This also indicates that only a subset of labels can be positive for an instance. It seems that there are blocks of positive labels in the data set (this is visualized by the heatmap of the data set in Figure 5.1a). The statistics also show that almost all instances have a distinct label combination. This makes it hard to learn meaningful models with certain multi-label classifiers such as label power set, described in Section 2.1.4.6.

An issue for multi-label classification on this data is the large number of missing values in the data set. Missing values in the labels have not yet been addressed in the multi-label literature. In classical multi-label applications, this is not very common[45]. In ToxCast™, 20% of the values are missing. The values are not missing at random, certain chemical classes were not tested for certain endpoints. Again, there is a pattern, which can be seen in the heatmap of missing values (see Figure 5.1b), the pattern of the missing values and the distribution of positive labels indicate that there are certain labels which might benefit from being predicted together using a multi-label classifier,

45 For example, tags of web sites, assays or pictures are simply present or not (0 or 1), but they are not missing.

whereas other labels not belonging to these blocks might not be beneficial for the model.

5.3 Method

In summary, one of the main issues in ToxCast™ might be the diverse quality of the endpoints. For some, there are not enough tested substances, for others, we observe skewed class distributions. Having only 309 compounds, makes it hard for the classifiers to perform well on all classes. Thus, before learning multi-label models on the endpoints, we introduce a filtering step to exclude labels which cannot be predicted well. Clearly this reduces the number of classes, but given the high number of endpoints, there will still exist many classes which can be predicted. Additionally, the number of useful labels will be improved with future releases of the data set. These releases will fill gaps in the data set, adding additional data for the labels.

The filtering step is done in the following way: First, a Binary Relevance (see Section2.1.4.1) classifier is learned and evaluated in a *leave-one-out cross-validation*. Next, the same is done with a more sophisticated multi-label classifier. In the experiments, we used ensembles of classifier chains (see Section 2.1.4.3) and multi-label classification using Boolean matrix decomposition (see Section 3.1.2). A Binary Relevance classifier does not exploit the label dependencies. The more sophisticated classifiers use the dependencies between the labels to improve the predictions. In the next step, we examine the performance of each label in both classification methods. If the performance of the Binary Relevance classifier is worse than the multi-label classifier, this label has dependencies to the other labels. Thus, it makes sense to use it in multi-label classification. The labels where the performance in both cases is the same or the binary relevance classifier outperforms the multi-label classifier do not benefit from multi-label classification. This means that in general they do not improve the overall performance of the prediction as there is no dependency from the other labels. Therefore, they are removed from the data set and a new multi-label model is learned by the multi-label classifier using only the labels which benefit from multi-label classification. To evaluate the performance on the filtered data set, a *leave-one-out cross-validation* is performed over the whole procedure. A detailed overview of the algorithm is given in Algorithm 5.1.

Applying the learned model uses the same procedure as for any other multi-label model. A new instance is passed to the model, and the model returns a bi-partition and a set of confidences for each label that was used in the final learning step. For the remaining labels, no prediction is returned.

Algorithm 5.1: Pseudo-code for training the ToxCast™ model

```
Input: ToxCast™ data set $\mathcal{D}$ with features $\mathcal{X}$ and labels $\mathcal{Y}$, selection threshold $\theta$
Output: Multi-label model $f(x), x \in \mathcal{X}$ and list of used endpoints $l$
/* empty set for predictions of Binary Relevance classifiers        */
$\hat{\mathcal{Y}}_{BR} \leftarrow \{\}$
/* empty set for predictions of multi-label classifier               */
$\hat{\mathcal{Y}}_{ML} \leftarrow \{\}$
/* perform n-fold cross-validation on the data set                   */
for $i \in 1, \ldots, n$ do
    /* generate training set $\mathcal{D}_{trg}$                      */
    $\mathcal{D}_{trg} \leftarrow train\_split(\mathcal{D}, i)$
    /* generate test sets consisting of features $\mathcal{X}_{trg}$ and labels $\mathcal{Y}_{trg}$ */
    $\mathcal{X}_{tst} \leftarrow test\_split(\mathcal{X}, i)$
    $\mathcal{Y}_{tst} \leftarrow test\_split(\mathcal{Y}, i)$
    /* train model $f_{BR}(x)$ with Binary Relevance Method           */
    $f_{BR} \leftarrow train_{BR}(\mathcal{D}_{trg})$
    /* train model $f_{ML}(x)$ with multi-label Method                */
    $f_{ML} \leftarrow train_{ML}(\mathcal{D}_{trg})$
    /* get predictions of the learned models                          */
    $\hat{\mathcal{Y}}_{BR_i} \leftarrow f_{BR}(\mathcal{X}_{tst})$
    $\hat{\mathcal{Y}}_{ML_i} \leftarrow f_{ML}(\mathcal{X}_{tst})$
    /* store predictions in global variable                           */
    $\hat{\mathcal{Y}}_{BR} \leftarrow add(\hat{\mathcal{Y}}_{BR}, \hat{\mathcal{Y}}_{BR_i})$
    $\hat{\mathcal{Y}}_{ML} \leftarrow add(\hat{\mathcal{Y}}_{ML}, \hat{\mathcal{Y}}_{ML_i})$
end
/* final set $\mathcal{Y}_{final}$ of labels used for training final model */
$\mathcal{Y}_{final} \leftarrow \{\}$
foreach $y_i \in \mathcal{Y}$ do
    /* calculate performance of classifiers on label $y_i$            */
    $perf_{BR_i} \leftarrow calculate\_performance(\hat{\lambda}_{BR_i}, \lambda_i)$
    $perf_{ML_i} \leftarrow calculate\_performance(\hat{\lambda}_{ML_i}, \lambda_i)$
    /* only add labels to final set that improve at least above a
       threshold when using multi-label classifiers                   */
    if $(perf_{ML_i} - perf_{BR_i}) > \theta$ then
        $\mathcal{Y}_{final} \leftarrow \mathcal{Y}_{final} \cup y_i$
    end
end
/* generate training set for final model                             */
$\mathcal{D}_{final} \leftarrow compose(\mathcal{X}, \mathcal{Y}_{final})$
/* learn final model                                                 */
$f(x) \leftarrow train_{ML}(\mathcal{D}_{final})$
```

5.3.1 Multi-Label Classification

In the previous section, we pointed out that it seems natural to use multi-label classification for the given task, building predictive models for ToxCast™. There are several algorithms which could be used in this case, with respect to the overview of multi-label learners given in Section 2.1. For the ToxCast™ data set, we decided to use MLC-BMaD (see Section 3.1.2) and Ensembles of Classifier Chains (see Section 2.1.4.3).

5.3.1.0.1 MLC-BMaD The big advantage of MLC-BMaD in this case is the compression of the label space. The ToxCast™ data set has a huge number of labels (and features) compared to other multi-label data sets (see Table 3.2). Therefore, the resources needed when using multi-label classifiers can be rather high, for example in BR2 (see Section 2.1.4.2), two classifiers need to be learned per label. Depending on the used base classifier, this can consume a lot of resources. By reducing the number of labels, this can be avoided. In our experiments, the algorithm usually reduced the number of labels to a number between 20 and 30. We used hill climbing to optimize the number of labels after the decomposition step.

Due to the missing values in the ToxCast™ data set, we enhanced the implementation to handle missing values. More precisely, the implementation used in previous experiments (see Section 3.1.2) uses an external library for the decomposition. Unfortunately, this library does not support missing values in the labels. Therefore, we used a new library implementing the same algorithm as before, but simply ignoring missing values. It was not necessary to change the algorithm of matrix decomposition itself, as it is capable of handling missing values.

5.3.1.0.2 Ensembles of Classifier Chains Ensembles of Classifier chains seem to be one of the best performing multi-label classifier at the moment. Compared to other classifiers, it is rather fast and uses only little resources, which is an important feature for the ToxCast™ data set. Only one single-label classifier is learned per label and ensemble. It performs well compared to other classifiers (see Section 2.1.4.3). As the classifier learns one label after the other and uses the previously learned model to predict the label, it is less affected by the missing labels. It simply does not use the missing information as input values for the base classifiers.

5.3.1.0.3 Base Classifier As a base classifier, we chose random forests [9]. They performed well without any parameter tuning in our experiments. While Support Vector Machines with optimized parameters can outperform random forests, in the case of multi-label classification, optimization of parameters on the level of base classifiers is expensive. Each single classifier needs to be optimized, slowing the learning process down as even more base classifiers have to be learned for each label. In most cases, parameter optimization requires an internal *cross-validation* using different parameter settings. This leads to a multiplication of the running time. Which is, in the case of ToxCast™, an even bigger issue: the running time even in the non-optimized base classifier setup was rather high and made the use of computing clusters necessary.

The base classifiers used in the experiments are rather simple. We did not use a special feature selection. While it seems to be fundamental for this data set to generate well performing classifiers [50, 3], the goal was to develop a method for selecting a group of endpoints and use this selection to improve the prediction for this group of endpoints. The overall results of this approach should improve after using more sophisticated base classifiers and preprocessing steps before classification[46].

5.4 Experimental Results

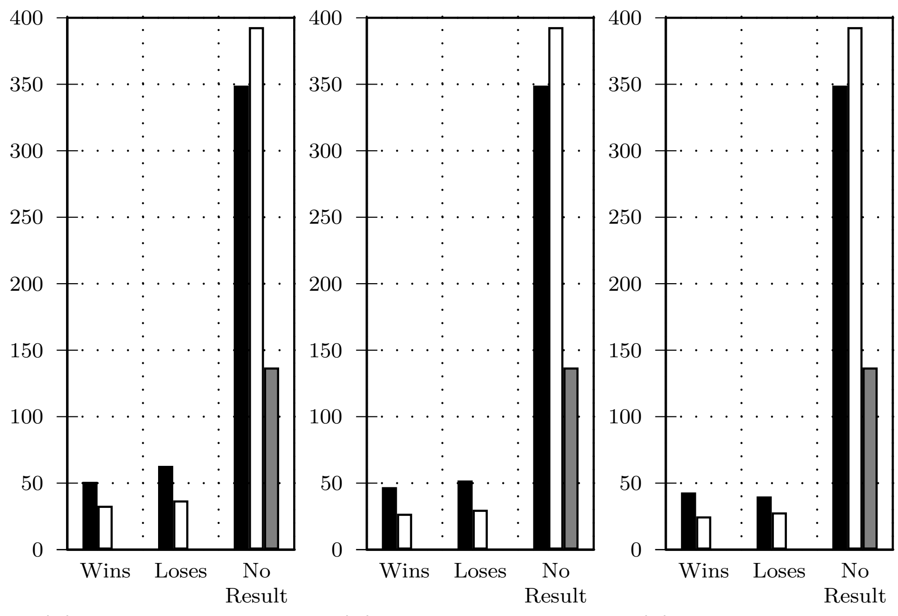
Evaluating the ToxCast™ data set, the most important part is to consider the performance on the single labels. The diverse quality of the data set makes it hard to get a good performance of any classifier on all labels. Hence, in the following, I will discuss and consider the performance on single labels instead of the overall performance on the data set. As the number of labels is too big to show the performance on all labels in this chapter, the detailed performance is given in Table A.2 in Appendix A.

Table 5.2 gives a selection of labels on which the proposed method performs particularly well. This table represents approximately 10% of the labels. It shows that the method can drastically improve the predictive performance on the single labels. Even though it is not possible to evaluate MLC-BMaD in many cases, it is shown that it can strongly improve the prediction of some labels, e.g., CHR_Rat_Bloodvessel_3_NeoplasticLesion or CHR_Rat_ZymbalsGland_3_NeoplasticLesion can benefit strongly, in both cases the area under curve is doubled compared to the baseline case of binary relevance. The performance gains using ECC in Table 5.2 are in most cases not as strong as for MLC-BMaD, but a broader range of labels can be covered.

46 It should be noted that random forests, used in the experiments, already conduct some feature selection. Therefore, a separate feature selection could only improve the performance when using different base classifiers, e.g. Support Vector Machines.

Table 5.2: List of labels on which multi-label classification and filtering performs particularly well. The selected labels are included if one of the multi-label classifiers performs strongly better than the other.

Label	BR	ECC	MLC-BMaD
CHR_Mouse_ArteryGeneral_2_PreneoplasticLesion	0.388	0.626	0.463
CHR_Mouse_BoneMarrow_2_PreneoplasticLesion	0.543	0.875	0.527
CHR_Mouse_Lung_1_AnyLesion	0.444	0.384	0.875
CHR_Mouse_MammaryGland_2_PreneoplasticLesion	0.415	0.531	0.9
CHR_Mouse_Nose_2_PreneoplasticLesion	0.569	0.28	0.764
CHR_Mouse_PaRathyroidGland_3_NeoplasticLesion	0.492	0.703	-
CHR_Mouse_Skin_2_PreneoplasticLesion	0.563	0.807	-
CHR_Mouse_ThyroidGland_1_AnyLesion	0.541	0.166	0.666
CHR_Mouse_Tooth_2_PreneoplasticLesion	0.544	0.875	-
CHR_Mouse_Uterus_3_NeoplasticLesion	0.614	0.777	-
CHR_Mouse_Vagina_1_AnyLesion	0.553	0.709	-
CHR_Rat_Bloodvessel_3_NeoplasticLesion	0.346	-	0.857
CHR_Rat_Brain_2_PreneoplasticLesion	0.405	-	0.833
CHR_Rat_HarderianGland_2_PreneoplasticLesion	0.495	0.602	0.583
CHR_Rat_IntestineSmall_2_PreneoplasticLesion	0.454	0.9	0.5
CHR_Rat_KidneyNephropathy	0.497	0.888	-
CHR_Rat_Nose_3_NeoplasticLesion	0.544	0.822	0.457
CHR_Rat_Penis_3_NeoplasticLesion	0.435	0.844	-
CHR_Rat_SeminalVesicle_2_PreneoplasticLesion	0.465	-	0.757
CHR_Rat_Tongue_1_AnyLesion	0.528	0.682	0.738
CHR_Rat_Tongue_2_PreneoplasticLesion	0.428	0.714	-
CHR_Rat_Trachea_1_AnyLesion	0.43	-	0.945
CHR_Rat_UncertainPrimarySite_3_NeoplasticLesion	0.562	0.5	0.727
CHR_Rat_UrinaryBladder_3_NeoplasticLesion	0.565	-	0.628
CHR_Rat_Uterus_1_AnyLesion	0.475	0.75	0.687
CHR_Rat_ZymbalsGland_3_NeoplasticLesion	0.416	0.115	0.846
DEV_Rabbit_Developmental	0.511	-	0.772
DEV_Rat_Developmental_Cardiovascular_MajorVessels	0.457	0.937	-
MGR_Rat_DevelopmentalLandmark	0.359	-	0.75



(a) Evaluation threshold 0.0 **(b)** Evaluation threshold 0.05 **(c)** Evaluation threshold 0.1

Figure 5.2: Number of wins and losses of ECC (black) and MLC-BMaD (white) over BR (gray) as well as number of no results. The bars indicate how many labels the given method performs at least the value of the evaluation threshold better than the BR method. The cases where no results where obtained in most cases is due to no positive predictions of the classifier and therefore the area under the curve cannot be calculated.

Figure 5.2 shows the number of wins and losses of ECC (black) and MLC-BMaD (white) over BR (gray), as well as the number of cases in which the evaluation did not succeed, which is mainly caused by too many unknown values in the label or a skewed class distribution, so that no performance could be calculated[47]. The figure shows that raising the evaluation threshold from 0.0 to 0.1 strongly reduces the number of losses of ECC or MLC-BMaD, while the number of wins shows only little change. This shows that the benefit from filtering the labels using the proposed approach, is rather strong, while the decrease in performance is rather weak.

Additionally, it seems that ECC is more suitable for the filtering than MLC-BMaD. ECC generally can be used in more cases and seems to win over the baseline case more often. On the other hand, MLC-BMaD increases the performance stronger than ECC.

47 In the case that there are no positive or no negative predictions for a label, which is not uncommon in ToxCast™, given the large number of missing values and the skewed class distribution.

ECC learns models similar to BR, the only difference is that it takes additional labels into account as features. Hence, it is more probable that it improves the performance for each label as it simply uses more features. On the other hand, MLC-BMaD can change its performance drastically taking away a label or adding an additional label. The matrix decomposition can produce completely different results, and small changes in the dependency matrix can drastically change the predictions.

Figure 5.3 and Figure 5.4 show the distribution of AUC values for the labels. The first figure gives the number of labels with a rounded AUC in 10 intervals. To better visualize the differences, Figure 5.4 gives the ratio of labels with the given AUC interval. The first thing to see is the high number of labels with an AUC between 0.4 and 0.6 when using the BR algorithm. Around 50% of all labels have an AUC around 0.4, which means a random performance. Altogether, approximately 90% of the labels predicted by BR have an AUC between 0.4 and 0.6. Only a small fraction of labels gain an AUC above 0.6 using binary relevance[48].

On the other hand, the multi-label classifiers give a more even distribution. The number of labels with an AUC between 0.3 and 0.6 are drastically reduced. At the same time, the number of labels with an AUC of 0.1 and 0.2 are increased (but staying still at a low level, around 10 of more than 400 labels), but a stronger increase in labels is achieved at the high performance values, between 0.7 and 0.9. Both classifiers can strongly increase the number of labels that are predicted with a high AUC, showing a reasonable to good classification performance.

MLC-BMaD stronger separates the labels into performing well and performing not so well, while ECC still has a higher range of random predictions for some labels. This complies with the expectation that small changes in the label space drastically change the Boolean matrix decomposition, which changes the performance in any direction, while small changes in the labels do not change the prediction of ECC that much, as the difference concerns only additional features of the base classifiers.

There is still a fraction of the labels with a low performance when using multi-label classifiers. However, this only applies to around 20 labels, which is less than 5% of the original labels, and only 20% of the used labels in the filtered predictions. At the same time, the prediction of around 25 labels is increased to a good performance and 210 labels with a performance at the same level as a random classifier are discarded.

48 However, this does not indicate that the labels cannot be predicted using binary classification, we used a basic algorithm and did not aim for the highest performance, but for improving any binary classification by the use of multi-label classification and filtering. Previous work has shown that well performing models can be trained for certain endpoints [21].

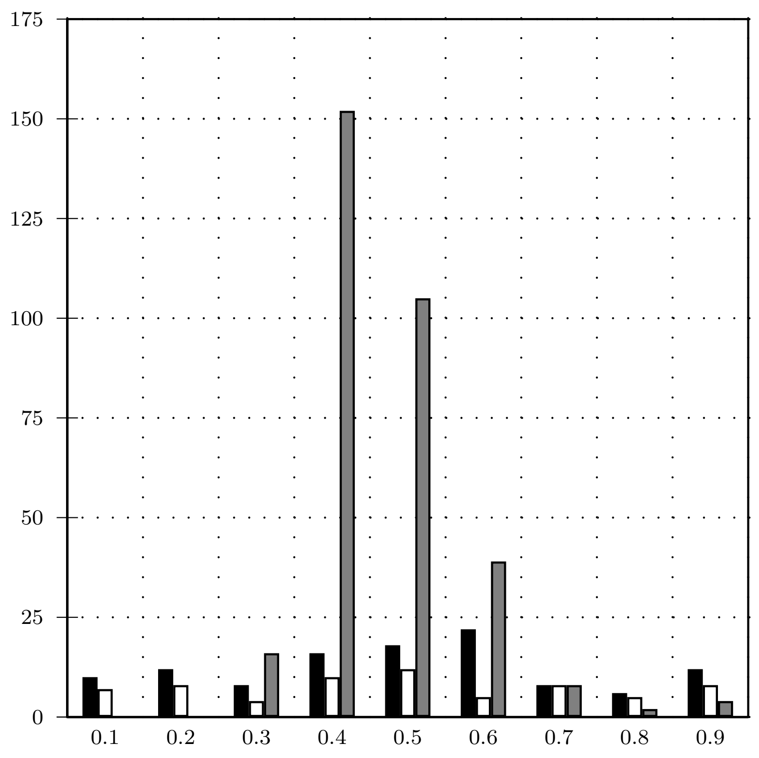


Figure 5.3: Distribution of the AUC values on the ToxCast™ data set for ensembles of classifier chains (black), MLC-BMaD (white) and BR (gray). Although BR predicts a lot more labels, most of the prediction performance is around or below random (0.5). On the other hand, the multi-label classifiers reduce the number of random predictions, yet they still exhibit bad performance.

Figure 5.5 gives a more detailed overview of the performance. The figure gives a graphical representation of the increase or decrease of performance in terms of area under the ROC curve. If a node A in the graph is above another node B, the method represented by A performs better on a label than the method represented by B. An edge gives the number of labels for which this relation holds true. In the center, the level of node BR represents the base line, the performance of the binary relevance algorithm. Any node above this level represents a performance higher than the baseline, any node below represents a performance lower than the baseline. The first layer to the left is the comparison method, binary relevance. The next layer represents the multi-label classifier without the filtering step, written as ML for multi-label classifier. In Figure 5.5a, the multi-label classifier is ensemble of classifier chains, in Figure 5.5b, the multi-

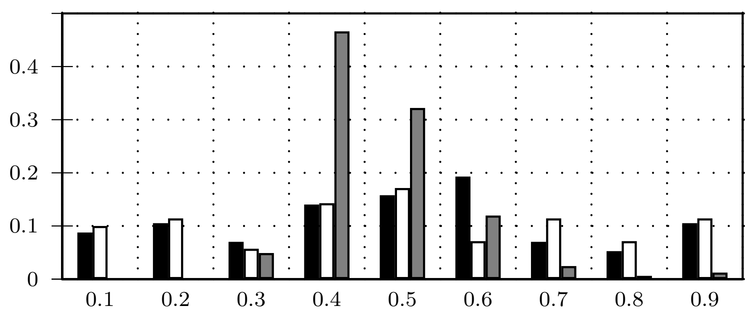


Figure 5.4: Relative distribution of the AUC values on the ToxCast™ data set for ensembles of classifier chains (black), MLC-BMaD (white) and BR (gray). The majority of the labels perform rather bad when using the BR classifier. When using a multi-label classifier, more labels get predicted well, with higher AUC values

label classifier is MLC-BMaD. The third layer, to the right, represents the filtered multi-label approach. Hence, an edge going from BR to FML_0 represents all labels that have a better performance with a filtered multi-label classifier (as the node FML_0 is higher than the node of BR) than with the binary relevance classifier, while at the same time no area under the curve could be calculated with a non-filtered multi-label classifier.

From this figure, more conclusions can be drawn. First, MLC-BMaD performs better on the complete data set than ECC. ECC performs worse than the base classifier in 157 labels, while MLC-BMaD only decreases the performance in about half of the cases, 89 labels, MLC-BMaD wins in 40 cases over BR, while ECC wins only in 27 cases. This complies with the observed behavior of MLC-BMaD, which performs particular well on data sets with a large number of labels. ToxCast™ consists of more than 400 labels, which is quite a lot compared to other multi-label data sets.

Second, ECC seems to generate a better filtering. In total, ECC filtering improves the performance in 68 cases[49], while it decreases the performance in 36 cases. Compared to MLC-BMaD, this is a strong improvement. Filtering using MLC-BMaD improves in the case of 28 labels and decreases the performance for 27 labels. On the other hand, when compared with the BR classifier, most of the improvements in the case of filtered ECC do not outperform the base method. 47 labels get improved above the base level, while for 48 labels the performance decreases. In case of MLC-BMaD, the results are

49 This can be easily extracted from the diagram by summing up all label numbers on edges pointing upwards to a FML_n node.

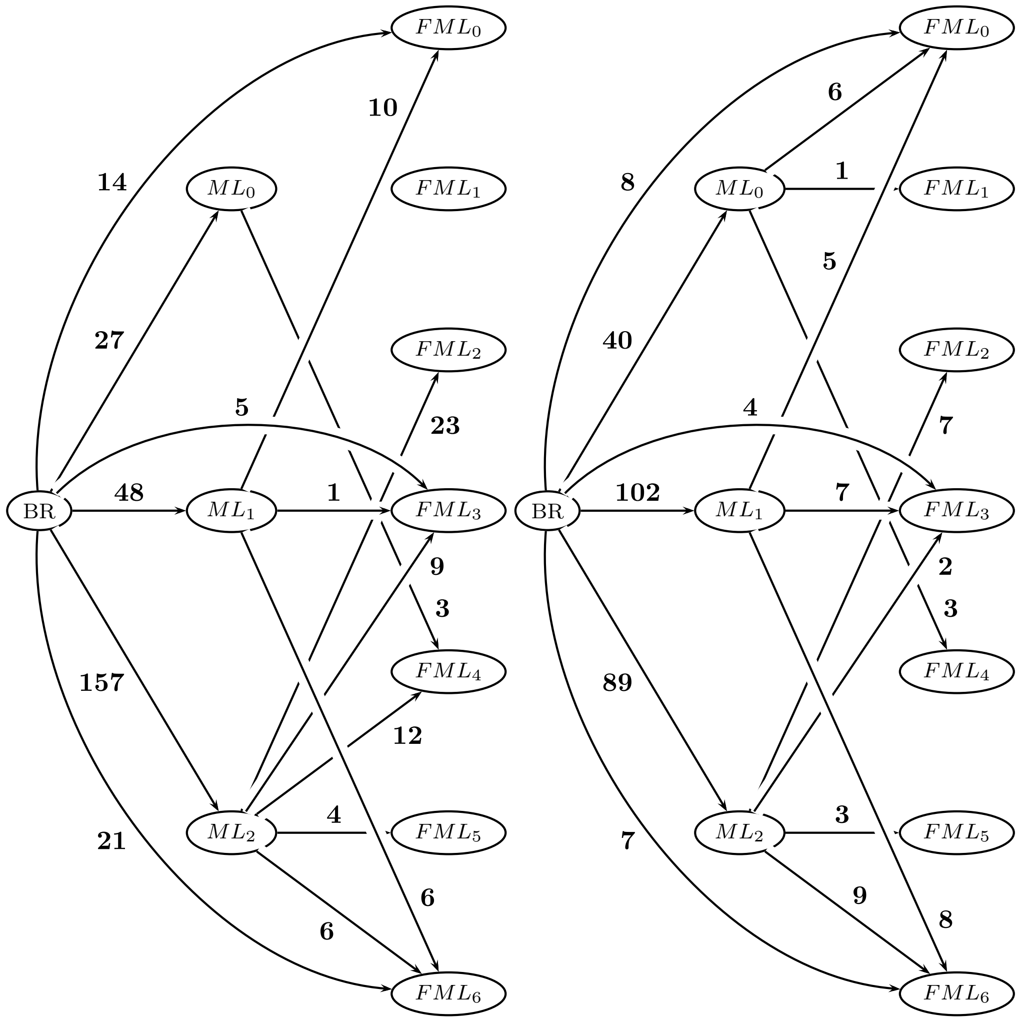
more stable, 27 labels improve the performance using the filtering, and 27 decrease the performance. This is also related to the slightly worse performance of the ECC algorithm in the unfiltered approach. This shows that it is harder for ECC to outperform BR on this data set (in the case of 157 labels, ECC loses to BR).

Third, both classifier can strongly reduce the number of labels in which the multi-label approach suffers from the diverse quality of the data set. ECC reduce the number of labels where it cannot outperform BR from 157 to 52, only one third of the labels remain. MLC-BMaD reduces the number of labels from 89 to 30 cases in which the multi-label approach performs worse than the base classifier.

To summarize, ECC has problems with learning well-performing models on the ToxCast™ data set, yet it seems to be a suitable for the filtering approach. On the other hand, MLC-BMaD performs better for multi-label classification compared to ECC, however, the filtering approach does not work that well using MLC-BMaD. This is attributed to the small number of labels which are present in the learning after the filtering. Previous experiments showed that MLC-BMaD strongly benefits from a large number of labels. However, both methods can filter a large fraction of labels where multi-label classification is not suitable due to missing information or a skewed class distribution.

These results show that using multi-label classification and filtering, even with a basic classifier like random forests with default parameters[50], certain labels can be used in classification, giving a high prediction performance in the evaluation. The baseline method using binary classification suffers from a large number of labels with random predictions.

50 Default parameters as set in the WEKA workbench [37].



(a) ECC **(b)** MLC-BMaD

Figure 5.5: Diagram of the performance of different experimental settings for the ToxCast™ data set. BR is the performance in terms of AUC of the baseline classifier (binary relevance), ML is the performance of the multi-label classifier, and FML the performance of the multi-label classifier using additional label filtering. If a node A is higher than another node B, i.e. closer to the root, the performance of the method given by A is better than the performance of the method given by B. The arrows indicate the number of labels that behave with the given performance change, the numbers give the number of labels with this change in performance. E.g., in Figure (a), 27 labels can be predicted at a higher performance with ECC than when using the BR classifier. Of these 27 labels, 3 perform worse than ECC and BR when using ECC with a filtering step. Edges pointing up represent an increase in performance, edges pointing down a decrease in performance. Only edges are given with a number of labels above 0, e.g., there are no labels with an improvement of the performance when using ECC, that have the same performance when using an additional filtering step. An increase or decrease in performance was only detected when the AUC differs at least 0.05.

5.5 Discussion and Future Work

This chapter introduced the use of multi-label classification for predictive toxicology by applying multi-label algorithms to the ToxCast™ data set. In addition to the multi-label classification, we implemented a filtering step to cope with the high number of labels with a low quality due to missing values or skewed class distributions. As multi-label classification has not been widely used in predictive toxicology, the introduction of these algorithms to the field is the main contribution of this chapter. The experiments showed the high potential for large classifier systems, caused by the type of data frequently found in predictive toxicology, a large number of labels combined with a low number of instances.

The experiments on the data set showed that there is a high potential for the proposed method. It can be used to prioritize endpoints for future releases of the data set. On the other hand, it improved the performance of a large subset of the labels compared to the baseline, the binary classification, which is used mostly in this field.

There are still some possibilities to improve the predictions. The used multi-label classifiers can be improved by using meta multi-label classifiers. Recent publications suggest that HOMER by Tsoumakas *et al.* [96] can improve classification performance for complex data sets (see also Section 2.1.4.8). HOMER groups the labels into subsets and uses other multi-label classifiers to train on these sets. This can be beneficial on data sets with many labels like ToxCast™. The dependencies in the grouped labels are used within the prediction. For data sets with a large number of labels, this can help the classifiers. Labels with no dependency to other labels can be grouped in different subgroups. Regarding our proposed method, this can lead to a larger number of labels in the final model. The set labels selected by the proposed method can then be grouped further into smaller sets, which is automatically done by HOMER.

Another reason for a performance drop of this method is due to the base classifier. We used Random Forest as they usually perform well and do not have a long training time, which is essential for large multi-label problems. The training of a model is multiplied by large factors depending on the number of labels, folds, and instances. A performance increase can be gained by using optimized Support Vector Machines. Nevertheless, this drastically increases the training time, making it hard to experiment with different settings. However, to obtain a model performing better than the best so far, it might be necessary to train parameter-optimized support vector machines.

CHAPTER 6

Conclusion

This thesis covers large classifier systems, which is an area of machine learning where multiple classifiers are combined into one system of classifiers. These systems are particularly useful for complex data, e.g., data which is distributed across multiple relations, or multiple target values, as in the case of multi-label classification. Two new classification methods for different scenarios are introduced, MLC-BMaD and ClassFact. In the following, multi-label classification was applied to two cheminformatics tasks. In the first, expert-based knowledge was combined with machine learning to predict the environmental fate of chemicals. The second application was the prediction of toxic effects of chemicals. This chapter will conclude the thesis by summarizing the previous chapters as well as the contributions of this thesis. It will finish with an outlook on future research on large classifier systems in bio- and cheminformatics.

Chapter 1 gave an introduction to the field of large classifier systems, which is a special case of classification in machine learning, where multiple classifiers are combined to predict target values that are structured in a more complex way than in the standard setting, where one target value is predicted by models given features in one relation. Large classifier systems cope with data with many target variables, as in the case of multi-label classification, or with target variables that are determined by data that is structured in multiple relations, as in the case of multi-relational learning. To motivate the presented work, I introduced two problems in the domain of biodegradation pathway prediction and predictive toxicology.

Next, Chapter 2 introduced the notation and measures for multi-label classification, which are used in the following chapters. Subsequently, I gave an overview of state-of-the-art algorithms using large classifier systems for multi-label classification. Multi-label classification extends the traditional machine learning classification task to multiple interdependent non-exclusive target values. Multi-label classification is the most common form of large classifier systems, as a large fraction of multi-label classifiers are based on transforming the data set into multiple binary classification tasks. For each new task, a new classifier is trained, hence the multi-label classifier is a large classifier system.

Next, I gave an introduction to multi-relational learning and the use of large classifier systems in this field. The use of classifier systems is rather uncommon in multi-relational learning. Yet it is a useful classification scheme for certain multi-relational problems, where data are stored in three main relations. This approach is closely related to collaborative filtering, which is often used in recommender systems [1].

The main part of the thesis is presented in Chapter 3, where two new approaches are introduced that use large classifier systems. Both approaches use Boolean matrix decomposition (BMD) to exploit dependencies between classes. The first approach, a multi-label algorithm called MLC-BMaD, factorizes the labels into two matrices. One stores the dependencies between the labels, the other the latent labels, which can be understood as representations of the original labels. The latent labels are predicted by the binary relevance classifier, as the latent labels are generated to be independent from each other, hence multi-label classifiers would not improve the prediction strongly. The presented algorithm, called MLC-BMaD, outperforms other algorithms in the evaluation. It performs particularly well when evaluated on multi-label data sets with a large number of labels compared to the number of instances. Hence, this classifier performs well when applied to multi-label data sets that really benefit from multi-label classification. When a data set is small in terms of the labels, the dependencies cannot be exploited by multi-label classifiers. When a data set is large in terms of the instances, there is enough information for each label and hence no additional information is required from other labels.

The second part of Chapter 3 extended MLC-BMaD towards multi-relational learning in a classifier called ClassFact. More precisely, BMD was used to factorize a matrix that represents class mappings of two sets of instances. This is closely related to two-way learning [72], which is used in recommender systems. As the matrix is factorized, one new relation is generated for each set of instances. Using them, multi-label algorithms can be used to learn models, which next can be used to predict the new labels for new combinations of instances. The final prediction of the class mapping is given by the Boolean multiplication of the predicted label sets. The evaluation of ClassFact is hard, due to the lack of available data sets. Although this may be a common problem, data is usually transformed to the propositional form and used in standard classification algorithms. Hence, the data cannot be used by ClassFact, as it is not easy to find this kind of data and transform it back to its original form.

After introducing new algorithms using large classifier systems, I applied large classifier systems to two cheminformatics problems in the subsequent chapters. In Chapter 4, I used large classifier systems to predict the environmental fate of chemicals, and in Chapter 5, I introduced large classifier systems to predict the toxicity of chemicals.

The task in biodegradation pathway prediction is to predict the degradation pathways of chemicals in the environment. All produced chemicals, e.g., drugs or cosmetics, sooner or later end in the environment. There, the compounds get transformed into new compounds, which might get transformed further and so forth. There are many potential biodegradation products for each chemical, hence a biodegradation pathway can be generated for each compound. Typically, these pathways are created using expert knowledge, like in the case of the UM-PPS system, where transformation rules are applied to compounds to generate potential products. In each step, multiple products are predicted, for each predicted product in turn, and so on. The rules only take the structure of the compound into account and hence are too general, which leads to an exponential growth of the biodegradation pathway. The products are predicted no matter if they occur in real-world transformations, the only condition is a mapping of the original structure to the left-hand side of a rule. Hence, methods are required to limit the results of the prediction. We proposed to use a system of classifiers, one for each rule, to predict the probability of that rule being correctly triggered for a compound. This approach was extended to use more sophisticated multi-label classifiers, i.e., ensembles of classifier chains and MLC-BMaD. The simple approach using single classifiers was able to predict meaningful results on a subset of 13 transformation rules, the remaining transformation rules did not provide enough data to generate meaningful models. The results showed that this approach is capable of reducing the number of products while not discarding too many correct products. The use of multi-label classifiers improved the overall performance even further and made the approach applicable to more transformation rules.

In Chapter 5, we applied large classifier systems to predictive toxicology, in this case, we used the ToxCast™ data set. Predictive toxicology seems to be a rewarding field for large classifier systems. More precisely, multi-label classifiers are suitable for classification problems with many interdependent, non-exclusive target values, with data sets with only few instances. In predictive toxicology, the task is to predict toxic effects of chemicals given their structure. As the experimental data that can be used for the training of models is expensive to generate, data typically consists of only a small number of instances (compounds). Additionally, as experiments are usually carried out in combination with other experiments, these data sets typically contain multiple target variables.

The ToxCast™ data set is an example of a predictive toxicology data set. It was released by the EPA in 2009 and consists of approximately 300 compounds mapped to more than 400 toxic effects. The main problem with this data set is the diverse quality of the target variables, many target variables have a lot of missing values, caused by different experimental settings and failed experiments. Another problem is the skewed class distribution in the labels. This leads to a mixed performance of the predictive models when using multi-label classification on the data set. We introduced a filtering step, in which we selected only labels that are suitable for multi-label classification and learned models on these subsets of labels. In this way, we were able to improve the performance on selected labels. This selection could, in principle, be used in future releases of the ToxCast™ data set to prioritize the targets.

6.1 Contributions

In the previous chapters, I gave an overview of the current state of large classifier systems, which are a useful tool for challenging real-world applications, especially in the life science. The main contribution to this field is the introduction of two new algorithms using large classifier systems together with Boolean matrix decomposition. The first approach uses Boolean matrix decomposition in multi-label classification, resulting in a high-quality classification algorithm. MLC-BMaD uses the properties of BMD and the factorized matrices to not only reduce the complexity of the learning problem, but also to exploit dependencies between the labels to gain high classification performance compared to other multi-label classification algorithms. The algorithm especially performs well on data sets with a high number of labels and high label cardinality. Hence, it performs well on data sets that are "genuine" multi-label data sets.

The second approach using BMD for large classifier systems is ClassFact, which extends MLC-BMaD to handle multi-relational data. ClassFact incorporates BMD to multi-relational learning and introduces a new learning scheme. Data in this scheme is usually merged into propositional form, loosing part of the structure and making the evaluation harder. Using ClassFact, dependencies between classes and instance combinations can be exploited and hence, prediction can be improved and evaluation of these tasks simplified.

These two approaches introduce Boolean matrix decomposition to classification. Although BMD has been known and used for some time, there was no learning algorithm that made use of BMD. ClassFact and MLC-BMaD introduce two ways of using the properties of a BMD for model training and classification.

On the application side, I presented two major applications for large classifier systems in cheminformatics. First, we introduced a mixed expert and machine learning-based approach for biodegradation pathway prediction. This approach uses the "best of both worlds" to predict the environmental fate of chemicals. So far, there exist only few computer aided prediction systems for biodegradation pathway prediction. Using the presented method, we enhance existing methods with machine learning and offer probability estimates for predicted products. This enables the prediction of far more correct degradation products, as the pathway can be generated further without false products causing a combinatorial explosion. Large classifier systems proved to improve the prediction of degradation products by enabling the prediction of far more transformation types.

Last, this thesis introduced large classifier systems to predictive toxicology. One major issue in predictive toxicology is the size and quality of the data sets. While data sets are usually covering only a small number of chemicals, and hence, consist of only few instances, the compounds are tested in parallel in multiple experiments. The endpoints of the experiments often share interdependencies. Nevertheless, the common approach in predictive toxicology is traditional single-label classification, which does not take valuable information about label dependencies into account. The experiments on the ToxCast$^{\text{TM}}$ data set proved that predictive toxicology can benefit from large classifier systems, in particular, multi-label classification.

6.2 Outlook

As the field of large classifier systems is rather young, there exist many ways where future research could proceed. The area of multi-label classification still lacks a sound theoretical basis that could be used to improve MLC-BMaD further. So far, due to the Boolean matrix decomposition algorithm, only linear label dependenicies are captured. Replacing Boolean matrix decomposition by methods like auto encoders [42], the classifier could be extended to the non-linear case.

Similar points apply to ClassFact. This algorithm could also benefit from the use of auto-encoders for certain data sets. Another extension to this method would be a modification of the base classifiers, which were random forests in the presented experiments. Nevertheless, they could be easily exchanged for multi-relational classifiers, and hence even more complex data with further linked relations could be supported by ClassFact.

On the application side, the prediction of biodegradation pathways offers a wide range of potential extensions. The presented approach predicts the degradation products of one single intput compound. The algorithm can easily be extended to predict complete pathways with accumulated predictions on each edge between a compound and its degradation product. Another step would be the use of more advanced input data. In recent years, due to the introduction of next generation sequencing (NGS) [62, 84], it became easy to sequence a vast number of organisms in a short prediod of time. As the possibility of degradation products depends on the environmental conditions and the microorganisms present when degradation takes place, NGS could provide genomic data on consortia of microorganisms, which then could be used in the prediction of biodegradation products in the given environment. The use of mass spectrometry is a similar extension. Mass spectrometry is a common tool for the prediction of biodegradation products. However, this is mostly used for manual annotation. Mass spectrometry can be used as input for the predictive models, which would improve the prediction. The problem with these enhancements is the lack of data, thus the first step would be collecting data on known products using mass spectrometry and NGS. An enhancement to biodegradation pathway prediction is the combination with predictive toxicology. The main targets when identifying degradation products are the ones that are toxic and hence a threat for humans and animals. Degradation prediction systems could be easily combined with predictive toxicology systems like OpenTox [39] to not only predict all potential degradation products, but to focus on those pathways producing the toxic chemicals.

The field of predictive toxicology has a high potential for large classifier systems. While there exist a lot of data sets in the right form to be used by large classifier systems, they were, to date, not used for classification so far. In the case of ToxCast™, the next phase will provide a larger data set, from which the trained models will benefit. The presented approach can suggest potential endpoints for future data sets. An enhancement to the presented approach would be the use of filtering mechanism and statistics on the labels to group them without the training of a model.

Given these conclusions and the research presented in the previous chapters, I believe that large classifier systems as presented in this thesis have a role to play in the future and upcoming research challenges in bio- and cheminformatics.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering 17*, 6 (2005), 734–749.

[2] Allwein, E. L., Schapire, R. E., and Singer, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research 1* (2001), 113–141.

[3] Almuallim, H., and Dietterich, T. G. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (1991), vol. 2, AAAI Press, pp. 547–552.

[4] Banerjee, A., and Ghosh, J. Scalable clustering algorithms with balancing constraints. *Data Mining and Knowledge Discovery 13*, 3 (2006), 365–395.

[5] Berg, E. L., Kunkel, E. J., Hytopoulos, E., and Plavec, I. Characterization of compound mechanisms and secondary activities by BioMAP analysis. *Journal of Pharmacological and Toxicological Methods 53*, 1 (2006), 67–74.

[6] Blockeel, H., and De Raedt, L. Top-down induction of first-order logical decision trees. *Artificial Intelligence 101*, 1 (1998), 285–297.

[7] Blockeel, H., Raedt, L. D., and Ramon, J. Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), Morgan Kaufmann, pp. 55–63.

[8] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. Learning multi-label scene classification. *Pattern Recognition 37*, 9 (2004), 1757–1771.

[9] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[10] BUCHWALD, F., RICHTER, L., AND KRAMER, S. Predicting a small molecule-kinase interaction map: A machine learning approach. *Journal of Cheminformatics 3*, 1 (2011), 1–17.

[11] BUTTON, W. G., JUDSON, P. N., LONG, A., AND VESSEY, J. D. Using absolute and relative reasoning in the prediction of the potential metabolism of xenobiotics. *Journal of Chemical Information and Computer Sciences 43*, 5 (2003), 1371–1377.

[12] CLARE, A., AND KING, R. D. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 42–53.

[13] CORTES, C., AND MOHRI, M. AUC optimization vs. error rate minimization. In *Proceedings of the 2003 Conference on Advances in Neural Information Processing Systems* (2004), vol. 16, pp. 313–320.

[14] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297.

[15] DAVIS, J., AND GOADRICH, M. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning* (2006), ACM, pp. 233–240.

[16] DEMBCZYNSKI, K., CHENG, W., AND HÜLLERMEIER, E. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning* (2010), pp. 279–286.

[17] DEMBCZYŃSKI, K., WAEGEMAN, W., CHENG, W., AND HÜLLERMEIER, E. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-Label Data* (2010), pp. 5–12.

[18] DIMITROV, S., KAMENSKA, V., WALKER, J., WINDLE, W., PURDY, R., LEWIS, M., AND MEKENYAN, O. Predicting the biodegradation products of perfluorinated chemicals using CATABOL. *SAR and QSAR in Environmental Research 15*, 1 (2004), 69–82.

[19] DIMITROV, S., PAVLOV, T., NEDELCHEVA, D., REUSCHENBACH, P., SILVANI, M., BIAS, R., COMBER, M., LOW, L., LEE, C., PARKERTON, T., ET AL. A kinetic model for predicting biodegradation. *SAR and QSAR in Environmental Research 18*, 5-6 (2007), 443–457.

[20] Dimou, A., Tsoumakas, G., Mezaris, V., Kompatsiaris, I., and Vlahavas, I. An empirical study of multi-label learning methods for video annotation. In *Proceeding of the 7th International Workshop on Content-Based Multimedia Indexing* (2009), IEEE, pp. 19–24.

[21] Dix, D. J., Houck, K. A., Judson, R. S., Kleinstreuer, N. C., Knudsen, T. B., Martin, M. T., Reif, D. M., Richard, A. M., Shah, I., Sipes, N. S., and Kavlock, R. J. Incorporating biological, chemical, and toxicological knowledge into predictive models of toxicity. *Toxicological Sciences 130*, 2 (2012), 440–441.

[22] Dix, D. J., Houck, K. A., Martin, M. T., Richard, A. M., Setzer, R. W., and Kavlock, R. J. The ToxCast program for prioritizing toxicity testing of environmental chemicals. *Toxicological Sciences 95*, 1 (2007), 5–12.

[23] Dong, L., Frank, E., and Kramer, S. Ensembles of balanced nested dichotomies for multi-class problems. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2005), Springer, pp. 84–95.

[24] Duygulu, P., Barnard, K., Freitas, J. d., and Forsyth, D. A. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision-Part IV* (2002), Springer-Verlag, pp. 97–112.

[25] Džeroski, S. Multi-relational data mining: an introduction. *SIGKDD Explorations Newsletter 5*, 1 (2003), 1–16.

[26] Elisseeff, A., and Weston, J. A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems 14* (2001), 681–687.

[27] Ellis, L. B., Roe, D., and Wackett, L. P. The University of Minnesota biocatalysis/biodegradation database: the first decade. *Nucleic Acids Research 34*, Database issue (2006), D517–D521.

[28] Fenner, K., Gao, J., Kramer, S., Ellis, L., and Wackett, L. Data-driven extraction of relative reasoning rules to limit combinatorial explosion in biodegradation pathway prediction. *Bioinformatics 24*, 18 (2008), 2079–2085.

[29] Frank, E., and Hall, M. A simple approach to ordinal classification. In *Proceedings of the 12th European Conference on Machine Learning* (2001), Springer, pp. 145–156.

[30] Freund, Y., and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 1 (1997), 119–139.

[31] Fröhler, S., and Kramer, S. Inductive logic programming for gene regulation prediction. *Machine Learning 70*, 2-3 (2008), 225–240.

[32] Fürnkranz, J., Hüllermeier, E., Mencía, E. L., and Brinker, K. Multilabel classification via calibrated label ranking. *Machine Learning 73*, 2 (2008), 133–153.

[33] Giuliano, K. A., Johnston, P. A., Gough, A., and Taylor, D. Systems cell biology based on high-content screening. *Methods in Enzymology 414* (2006), 601–619.

[34] Godbole, S., and Sarawagi, S. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[35] Gómez, M. J., Pazos, F., Guijarro, F. J., de Lorenzo, V., and Valencia, A. The environmental fate of organic pollutants through the global microbial metabolism. *Molecular Systems Biology 3*, 1 (2007), 299–314.

[36] Greene, N., Judson, P., Langowski, J., and Marchant, C. Knowledge-based expert systems for toxicity and metabolism prediction: DEREK, StAR and METEOR. *SAR and QSAR in Environmental Research 10*, 2-3 (1999), 299–314.

[37] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 10–18.

[38] Hansch, C., Leo, A., Hoekman, D., and Heller, S. R. *Exploring QSAR*. American Chemical Society, 1995.

[39] Hardy, B., Douglas, N., Helma, C., Rautenberg, M., Jeliazkova, N., Jeliazkov, V., Nikolova, I., Benigni, R., Tcheremenskaia, O., Kramer, S., Girschick, T., Buchwald, F., Wicker, J., Karwath, A., Gütlein, M.,

Maunz, A., Sarimveis, H., Melagraki, G., Afantitis, A., Sopasakis, P., Gallagher, D., Poroikov, V., Filimonov, D., Zakharov, A., Langunin, A., Gloriozova, T., Novikov, S., Skvortsova, N., Druzhilovsky, D., Chawla, S., Gosh, I., Ray, S., Patel, H., and Escher, S. Collaborative development of predictive toxicology applications. *Journal of Cheminformatics 2*, 7 (2010), 1–29.

[40] Hastie, T., and Tibshirani, R. Classification by pairwise coupling. *The Annals of Statistics 26*, 2 (1998), 451–471.

[41] Helma, C., Ed. *Predictive toxicology.* CRC Press, 2005.

[42] Hinton, G. E., Revow, M., and Dayan, P. Recognizing handwritten digits using mixtures of linear models. *Advances in Neural Information Processing Systems* (1995), 1015–1022.

[43] Hou, B. K., Ellis, L. B., and Wackett, L. P. Encoding microbial metabolic logic: predicting biodegradation. *Journal of Industrial Microbiology and Biotechnology 31*, 6 (2004), 261–272.

[44] Hsu, C.-W., and Lin, C.-J. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks 13*, 2 (2002), 415–425.

[45] Hsu, D., Kakade, S., Langford, J., and Zhang, T. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Neural Information Processing Systems Foundation, 2009, pp. 772–780.

[46] Huang, R., Xia, M., Cho, M.-H., Sakamuru, S., Shinn, P., Houck, K. A., Dix, D. J., Judson, R. S., Witt, K. L., Kavlock, R. J., Tice, R. R., and Austin, C. P. Chemical genomics profiling of environmental chemical modulation of human nuclear receptors. *Environmental Health Perspectives 119*, 8 (2011), 1142–1148.

[47] Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. Label ranking by learning pairwise preferences. *Artificial Intelligence 172*, 16 (2008), 1897–1916.

[48] Inglese, J., Auld, D. S., Jadhav, A., Johnson, R. L., Simeonov, A., Yasgar, A., Zheng, W., and Austin, C. P. Quantitative high-throughput

screening: a titration-based approach that efficiently identifies biological activities in large chemical libraries. *Proceedings of the National Academy of Sciences 103*, 31 (2006), 11473–11478.

[49] Joachims, T., Hofmann, T., Yue, Y., and Yu, C.-N. Predicting structured objects with support vector machines. *Communications of the ACM 52*, 11 (2009), 97–104.

[50] Judson, R., Elloumi, F., Setzer, R. W., Li, Z., and Shah, I. A comparison of machine learning algorithms for chemical toxicity classification using a simulated multi-scale data model. *BMC Bioinformatics 9*, 1 (2008), 241–257.

[51] Judson, R., Richard, A., Dix, D. J., Houck, K., Martin, M., Kavlock, R., Dellarco, V., Henry, T., Holderman, T., Sayre, P., Tan, S., Carpenter, T., and Edwin, S. The toxicity data landscape for environmental chemicals. *Environmental Health Perspectives 117*, 5 (2009), 685–695.

[52] Judson, R. S., Houck, K. A., Kavlock, R. J., Knudsen, T. B., Martin, M. T., Mortensen, H. M., Reif, D. M., Rotroff, D. M., Shah, I., Richard, A. M., and Dix, D. J. In vitro screening of environmental chemicals for targeted testing prioritization: the ToxCast project. *Environmental Health Perspectives 118*, 4 (2010), 485–492.

[53] Judson, R. S., Martin, M. T., Egeghy, P., Gangwal, S., Reif, D. M., Kothiya, P., Wolf, M., Cathey, T., Transue, T., Smith, D., et al. Aggregating data for computational toxicology applications: the US environmental protection agency (EPA) aggregated computational toxicology resource (ACtoR) system. *International Journal of Molecular Sciences 13*, 2 (2012), 1805–1831.

[54] Katakis, I., Tsoumakas, G., and Vlahavas, I. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challlenge* (2008), pp. 75–83.

[55] Kavlock, R., Chandler, K., Houck, K., Hunter, S., Judson, R., Kleinstreuer, N., Knudsen, T., Martin, M., Padilla, S., Reif, D., Richard, A. M., Rotroff, D. M., Sipes, N. S., and Dix, D. J. Update on EPA's ToxCast program: Providing high throughput decision support tools for chemical risk management. *Chemical Research in Toxicology 25*, 7 (2012), 1287–302.

[56] Kleinstreuer, N. C., Judson, R. S., Reif, D. M., Sipes, N. S., Singh, A. V., Chandler, K. J., DeWoskin, R., Dix, D. J., Kavlock, R. J., and Knudsen, T. B. Environmental impact on vascular development predicted by high-throughput screening. *Environmental Health Perspectives 119*, 11 (2011), 1596–1603.

[57] Klopman, G., Tu, M., and Talafous, J. META 3 a genetic algorithm for metabolic transform priorities optimization. *Journal of Chemical Information and Computer Sciences 37*, 2 (1997), 329–334.

[58] Knudsen, T. B., Houck, K. A., Sipes, N. S., Singh, A. V., Judson, R. S., Martin, M. T., Weissman, A., Kleinstreuer, N. C., Mortensen, H. M., Reif, D. M., Rabinowitza, J. R., Setzera, R. W., Richarda, A. M., Dixa, D. J., and Kavlockaothers, R. J. Activity profiles of 309 ToxCast™ chemicals evaluated across 292 biochemical targets. *Toxicology 282*, 1 (2011), 1–15.

[59] Knudsen, T. B., Martin, M. T., Kavlock, R. J., Judson, R. S., Dix, D. J., and Singh, A. V. Profiling the activity of environmental chemicals in prenatal developmental toxicity studies using the US EPA's ToxRefDB. *Reproductive Toxicology 28*, 2 (2009), 209–219.

[60] Kramer, S., Widmer, G., Pfahringer, B., and De Groeve, M. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae 47*, 1 (2001), 1–13.

[61] Liu, Y., Jin, R., and Yang, L. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *Proceedings of the National Conference on Artificial Intelligence* (2006), vol. 21, pp. 421–426.

[62] Mardis, E. R. The impact of next-generation sequencing technology on genetics. *Trends in Genetics 24*, 3 (2008), 133–141.

[63] Martin, M. T., Dix, D. J., Judson, R. S., Kavlock, R. J., Reif, D. M., Richard, A. M., Rotroff, D. M., Romanov, S., Medvedev, A., Poltoratskaya, N., Gambarian, M., Moeser, M., Makarov, S., and Houck, K. Impact of environmental chemicals on key transcription regulators and correlation to toxicity end points within EPA's ToxCast program. *Chemical Research in Toxicology 23*, 3 (2010), 578–590.

[64] Martin, M. T., Judson, R. S., Reif, D. M., Kavlock, R. J., and Dix, D. J. Profiling chemicals based on chronic toxicity results from the US EPA ToxRef database. *Environmental Health Perspectives 117*, 3 (2009), 392–399.

[65] Martin, M. T., Knudsen, T. B., Reif, D. M., Houck, K. A., Judson, R. S., Kavlock, R. J., and Dix, D. J. Predictive model of rat reproductive toxicity from ToxCast high throughput screening. *Biology of Reproduction 85*, 2 (2011), 327–339.

[66] Martin, M. T., Mendez, E., Corum, D. G., Judson, R. S., Kavlock, R. J., Rotroff, D. M., and Dix, D. J. Profiling the reproductive toxicity of chemicals from multigeneration studies in the toxicity reference database (ToxRefDB). *Toxicological Sciences 110*, 1 (2009), 181–190.

[67] Mencia, E. L., and Fürnkranz, J. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 50–65.

[68] Miettinen, P. The boolean column and column-row matrix decompositions. *Data Mining and Knowledge Discovery 17*, 1 (2008), 39–56.

[69] Mitchell, T. M. *Machine learning*. McGraw-Hill, 1997.

[70] Mu, F., Unkefer, P. J., Unkefer, C. J., and Hlavacek, W. S. Prediction of oxidoreductase-catalyzed reactions based on atomic properties of metabolites. *Bioinformatics 22*, 24 (2006), 3082–3088.

[71] Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D., Johnson, N., Cohen, K. B., and Duch, W. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing* (2007), Association for Computational Linguistics, pp. 97–104.

[72] Piccart, B., Blockeel, H., Georges, A., and Eeckhout, L. Predictive learning in two-way datasets. http://ilp11.doc.ic.ac.uk/short_papers/ilp2011_submission_26.pdf, 2012.

[73] Platt, J. C. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999, pp. 185–208.

[74] Quinlan, J. R. *C4.5: programs for machine learning*, vol. 1. Morgan Kaufmann, 1993.

[75] REACH. Regulation (EC) no 1907/2006 of the European Parliament and of the council of 18 December 2006 concerning the registration, evaluation, authorisation and restriction of chemicals (REACH). *Official Journal of the European Union 49* (2006), L396.

[76] Read, J. *Scalable Multi-Label Classification.* PhD thesis, Department of Computer Science, University of Waikato, 2010.

[77] Read, J., Pfahringer, B., and Holmes, G. Multi-label classification using ensembles of pruned sets. In *8th IEEE International Conference on Data Mining* (2008), IEEE, pp. 995–1000.

[78] Read, J., Pfahringer, B., Holmes, G., and Frank, E. Classifier chains for multi-label classification. In *Machine Learning and Knowledge Discovery in Databases.* Springer, 2009, pp. 254–269.

[79] Richter, L., Hechtl, S., and Kramer, S. Leveraging chemical background knowledge for the prediction of growth inhibition. In *Proceedings of the 6th IEEE Symposium on BionInformatics and BioEngineering* (2006), IEEE, pp. 319–324.

[80] Rotroff, D. M., Beam, A. L., Dix, D. J., Farmer, A., Freeman, K. M., Houck, K. A., Judson, R. S., LeCluyse, E. L., Martin, M. T., Reif, D. M., and Fergusonc, S. S. Xenobiotic-metabolizing enzyme and transporter gene expression in primary cultures of human hepatocytes modulated by ToxCast chemicals. *Journal of Toxicology and Environmental Health 13*, 2-4 (2010), 329–346.

[81] Rückert, U., and Kramer, S. Frequent free tree discovery in graph data. In *Proceedings of the 2004 ACM Symposium on Applied Computing* (2004), ACM, pp. 564–570.

[82] Schapire, R. E., and Freund, Y. *Boosting: Foundations and Algorithms.* MIT Press, 2012.

[83] Schapire, R. E., and Singer, Y. BoosTexter: A boosting-based system for text categorization. *Machine Learning 39*, 2-3 (2000), 135–168.

[84] Shendure, J., and Ji, H. Next-generation DNA sequencing. *Nature Biotechnology 26*, 10 (2008), 1135–1145.

[85] Shukla, S. J., Huang, R., Austin, C. P., and Xia, M. The future of toxicity testing: a focus on in vitro methods using a quantitative high-throughput screening platform. *Drug Discovery Today 15*, 23 (2010), 997–1007.

[86] Sinclair, C. J., and Boxall, A. B. Assessing the ecotoxicity of pesticide transformation products. *Environmental Science & Technology 37*, 20 (2003), 4617–4625.

[87] Sipes, N. S., Martin, M. T., Reif, D. M., Kleinstreuer, N. C., Judson, R. S., Singh, A. V., Chandler, K. J., Dix, D. J., Kavlock, R. J., and Knudsen, T. B. Predictive models of prenatal developmental toxicity from ToxCast high-throughput screening data. *Toxicological Sciences 124*, 1 (2011), 109–127.

[88] Struyf, J., Džeroski, S., Blockeel, H., and Clare, A. Hierarchical multi-classification with predictive clustering trees in functional genomics. In *Progress in Artificial Intelligence*, C. Bento, A. Cardoso, and G. Dias, Eds., vol. 3808 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 2005, pp. 272–283.

[89] Tai, F., and Lin, H.-T. Multi-label classification with principle label space transformation. In *Proceedings of the 2nd International Workshop on Learning from Multi-Label Data* (2010), G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, Eds., pp. 45–52.

[90] Tax, D. M., and Duin, R. P. Using two-class classifiers for multiclass classification. In *Proceedings of the 16th International Conference on Pattern Recognition* (2002), vol. 2, IEEE, pp. 124–127.

[91] Tenenboim-Chekina, L., Rokach, L., and Shapira, B. Identification of label dependencies for multi-label classification. In *Proceedings of the 2nd International Workshop on Learning from Multi-Label Data* (2010), G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, Eds., pp. 53–60.

[92] Thomas, R. S., Black, M. B., Li, L., Healy, E., Chu, T.-M., Bao, W., Andersen, M. E., and Wolfinger, R. D. A comprehensive statistical analysis of predicting in vivo hazard using high-throughput in vitro screening. *Toxicological Sciences 128*, 2 (2012), 398–417.

[93] Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I., and Vlahavas, I. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceeding of ECML/PKDD 2009 Workshop on Learning from Multi-Label Data* (2009), G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, Eds., pp. 101–116.

[94] Tsoumakas, G., and Katakis, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining 3*, 3 (2007), 1–13.

[95] Tsoumakas, G., Katakis, I., and Vlahavas, I. A review of multi-label classification methods. In *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery* (2006), pp. 99–109.

[96] Tsoumakas, G., Katakis, I., and Vlahavas, I. Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data* (2008), pp. 30–44.

[97] Tsoumakas, G., Katakis, I., and Vlahavas, I. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook.* Springer, 2010, pp. 667–685.

[98] Tsoumakas, G., and Vlahavas, I. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning* (2007), Springer, pp. 406–417.

[99] Tsoumakas, G., Zhang, M.-L., and Zhou, Z.-H., Eds. *Proceedings of the First International Workshop on Learning from Multi-Label Data* (2009).

[100] Tsoumakas, G., Zhang, M.-L., and Zhou, Z.-H., Eds. *Proceedings of the Second International Workshop on Learning from Multi-Label Data* (2010).

[101] Tsoumakas, K. T. G., Kalliris, G., and Vlahavas, I. Multi-label classification of music into emotions. In *Proceedings of the 9th International Conference of Music Information Retrieval* (2008), p. 325.

[102] Turnbull, D., Barrington, L., Torres, D., and Lanckriet, G. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing 16*, 2 (2008), 467–476.

[103] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. Decision trees for hierarchical multi-label classification. *Machine Learning 73*, 2 (2008), 185–214.

[104] Wicker, J., Fenner, K., Ellis, L., Wackett, L., and Kramer, S. Machine learning and data mining approaches to biodegradation pathway prediction. In *Proceedings of the 2nd International Workshop on the Induction of Process Models at ECML PKDD 2008* (2008), W. Bridewell, T. Calders, A. K. de Medeiros, S. Kramer, M. Pechenizkiy, and L. Todorovski, Eds.

[105] Wicker, J., Fenner, K., Ellis, L., Wackett, L., and Kramer, S. Predicting biodegradation products and pathways: a hybrid knowledge- and machine learning-based approach. *Bioinformatics 26*, 6 (2010), 814–821.

[106] Wicker, J., Pfahringer, B., and Kramer, S. Multi-label classification using Boolean matrix decomposition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), ACM, pp. 179–186.

[107] Zhang, M.-L., and Zhou, Z.-H. A k-nearest neighbor based algorithm for multi-label classification. In *IEEE International Conference on Granular Computing* (2005), vol. 2, IEEE, pp. 718–721.

[108] Zhang, M.-L., and Zhou, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering 18*, 10 (2006), 1338–1351.

[109] Zhang, M.-L., and Zhou, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition 40*, 7 (2007), 2038–2048.

Appendices

APPENDIX A

ToxCast

A.1 Chemicals

Table A.1: List of chemicals from phase I of ToxCast™. ID, CASRN and Name is given. 309 chemicals are publicly available in phase I. The chemicals are all environmental chemicals. The majority of the chemicals in ToxCast™ are registered active pesticides (273 of the 309 structures). Only 18 chemicals are de-registered active pesticides, 22 are inert pesticides and 33 antimicrobials.

GCID	CASRN	Name
16199	111812-58-9	(Z,E)-Fenpyroximate
16069	2971-36-0	2,2-Bis(4-hydroxyphenyl)-1,1,1-trichloroethane (HPTE)
16071	94-82-6	2,4-DB
16070	94-75-7	2,4-Dichlorophenoxyacetic acid (2,4-D)
16072	136-45-8	2,5-Pyridinedicarboxylic acid, dipropyl ester
16073	90-43-7	2-Phenylphenol
16074	55406-53-6	3-Iodo-2-propynylbutylcarbamate
16075	1007-28-9	6-Deisopropylatrazine
16076	71751-41-2	Abamectin
16077	30560-19-1	Acephate
16078	135410-20-7	Acetamiprid
16079	34256-82-1	Acetochlor
16080	135158-54-2	Acibenzolar-S-Methyl
16081	50594-66-6	Acifluorfen
16082	15972-60-8	Alachlor
16083	116-06-3	Aldicarb

(continued)

GCID	CASRN	Name
16084	834-12-8	Ametryn
16085	33089-61-1	Amitraz
16086	101-05-3	Anilazine
16087	3337-71-1	Asulam
16088	1912-24-9	Atrazine
16089	35575-96-3	Azamethiphos
16090	86-50-0	Azinphos-methyl
16091	131860-33-8	Azoxystrobin
16092	22781-23-3	Bendiocarb
16093	1861-40-1	Benfluralin
16094	17804-35-2	Benomyl
16095	83055-99-6	Bensulfuron-methyl
16096	741-58-2	Bensulide
16097	25057-89-0	Bentazone
16098	149877-41-8	Bifenazate
16099	82657-04-3	Bifenthrin
16100	80-05-7	Bisphenol A
16101	10043-35-3	Boric acid
16102	188425-85-6	Boscalid
16103	314-40-9	Bromacil
16104	1689-84-5	Bromoxynil
16105	69327-76-0	Buprofezin
16106	23184-66-9	Butachlor
16107	134605-64-4	Butafenacil
16108	33629-47-9	Butralin
16109	2008-41-5	Butylate
16110	75-60-5	Cacodylic acid
16124	2425-06-1	Captafol
16111	133-06-2	Captan
16112	63-25-2	Carbaryl
16113	5234-68-4	Carboxin
16114	128639-02-1	Carfentrazone-ethyl
16115	54593-83-8	Chlorethoxyfos

(continued)

GCID	CASRN	Name
16116	1698-60-8	Chloridazon
16117	2675-77-6	Chloroneb
16118	1897-45-6	Chlorothalonil
16119	101-21-3	Chlorpropham
16120	5598-15-2	Chlorpyrifos oxon
16121	5598-13-0	Chlorpyrifos-methyl
16122	64902-72-3	Chlorsulfuron
16123	87818-31-3	Cinmethylin
16125	105512-06-9	Clodinafop-propargyl
16126	74115-24-5	Clofentezine
16127	81777-89-1	Clomazone
16128	101-10-0	Cloprop
16129	1702-17-6	Clopyralid
16130	57754-85-5	Clopyralid-olamine
16131	120-32-1	Clorophene
16132	210880-92-5	Clothianidin
16133	56-72-4	Coumaphos
16134	420-04-2	Cyanamide
16135	21725-46-2	Cyanazine
16136	120116-88-3	Cyazofamid
16137	113136-77-9	Cyclanilide
16138	1134-23-2	Cycloate
16139	68359-37-5	Cyfluthrin
16140	122008-85-9	Cyhalofop-butyl
16141	57966-95-7	Cymoxanil
16142	52315-07-8	Cypermethrin
16143	94361-06-5	Cyproconazole
16144	121552-61-2	Cyprodinil
16145	66215-27-8	Cyromazine
16146	1596-84-5	Daminozide
16147	533-74-4	Dazomet
16148	584-79-2	d-cis,trans-Allethrin
16149	333-41-5	Diazinon

(continued)

GCID	CASRN	Name
16150	962-58-3	Diazoxon
16151	84-74-2	Dibutyl phthalate
16152	1918-00-9	Dicamba
16153	1194-65-6	Dichlobenil
16154	99-30-9	Dichloran
16155	120-36-5	Dichlorprop
16156	62-73-7	Dichlorvos
16157	51338-27-3	Diclofop-methyl
16158	145701-21-9	Diclosulam
16159	115-32-2	Dicofol
16160	141-66-2	Dicrotophos
16162	117-81-7	Diethylhexyl phthalate
16163	134-62-3	Diethyltoluamide
16164	119446-68-3	Difenoconazole
16165	43222-48-6	Difenzoquat metilsulfate
16166	87674-68-8	Dimethenamid
16167	60-51-5	Dimethoate
16168	110488-70-5	Dimethomorph
16169	131-11-3	Dimethyl phthalate
16170	83657-24-3	Diniconazole
16171	122-39-4	Diphenylamine
16172	85-00-7	Diquat dibromide
16173	298-04-4	Disulfoton
16174	97886-45-8	Dithiopyr
16175	330-54-1	Diuron
16177	155569-91-8	Emamectin benzoate
16178	115-29-7	Endosulfan
16179	759-94-4	EPTC
16180	66230-04-4	Esfenvalerate
16181	55283-68-6	Ethalfluralin
16182	97780-06-8	Ethametsulfuron methyl
16183	16672-87-0	Ethephon
16184	26225-79-6	Ethofumesate

(continued)

GCID	CASRN	Name
16185	13194-48-4	Ethoprop
16186	96-45-7	Ethylenethiourea
16187	153233-91-1	Etoxazole
16188	2593-15-9	Etridiazole
16189	131807-57-3	Famoxadone
16190	161326-34-7	Fenamidone
16191	22224-92-6	Fenamiphos
16192	60168-88-9	Fenarimol
16193	114369-43-6	Fenbuconazole
16194	126833-17-8	Fenhexamid
16195	122-14-5	Fenitrothion
16196	66441-23-4	Fenoxaprop-ethyl
16197	72490-01-8	Fenoxycarb
16198	39515-41-8	Fenpropathrin
16200	55-38-9	Fenthion
16201	76-87-9	Fentin
16202	120068-37-3	Fipronil
16203	69806-50-4	Fluazifop-butyl
16204	79241-46-6	Fluazifop-P-butyl
16205	79622-59-6	Fluazinam
16206	131341-86-1	Fludioxonil
16207	142459-58-3	Flufenacet
16208	188489-07-8	Flufenpyr-ethyl
16209	62924-70-3	Flumetralin
16210	98967-40-9	Flumetsulam
16211	87546-18-7	Flumiclorac-pentyl
16212	103361-09-7	Flumioxazin
16213	2164-17-2	Fluometuron
16214	361377-29-9	Fluoxastrobin
16215	69377-81-7	Fluroxypyr
16216	81406-37-3	Fluroxypyr-meptyl
16217	85509-19-9	Flusilazole
16218	117337-19-6	Fluthiacet-methyl

(continued)

GCID	CASRN	Name
16219	66332-96-5	Flutolanil
16220	133-07-3	Folpet
16221	173159-57-4	Foramsulfuron
16222	68157-60-8	Forchlorfenuron
16223	23422-53-9	Formetanate hydrochloride
16224	98886-44-3	Fosthiazate
16225	100784-20-1	Halosulfuron-methyl
16226	79983-71-4	Hexaconazole
16227	51235-04-2	Hexazinone
16228	78587-05-0	Hexythiazox
16229	119515-38-7	Icaridin
16230	35554-44-0	Imazalil
16231	114311-32-9	Imazamox
16232	104098-48-8	Imazapic
16233	81334-34-1	Imazapyr
16234	81335-37-7	Imazaquin
16235	81335-77-5	Imazethapyr
16236	138261-41-3	Imidacloprid
16237	173584-44-6	Indoxacarb
16238	144550-36-7	Iodosulfuron-methyl-sodium
16239	36734-19-7	Iprodione
16240	42509-80-8	Isazofos
16241	82558-50-7	Isoxaben
16242	141112-29-0	Isoxaflutole
16243	77501-63-4	Lactofen
16244	58-89-9	Lindane
16245	330-55-2	Linuron
16246	1634-78-2	Malaoxon
16247	121-75-5	Malathion
16248	123-33-1	Maleic hydrazide
16249	8018-01-7	Mancozeb
16250	12427-38-2	Maneb
16251	94-74-6	MCPA

(continued)

GCID	CASRN	Name
16252	24307-26-4	Mepiquat chloride
16253	208465-21-8	Mesosulfuron-methyl
16254	104206-82-8	Mesotrione
16255	57837-19-1	Metalaxyl
16256	6734-80-1	Metam-sodium hydrate
16257	10265-92-6	Methamidophos
16258	950-37-8	Methidathion
16259	16752-77-5	Methomyl
16260	72-43-5	Methoxychlor
16261	161050-58-4	Methoxyfenozide
16262	109-86-4	Methyl cellusolve
16263	4376-18-5	Methyl hydrogen phthalate
16264	556-61-6	Methyl isothiocyanate
16265	6317-18-6	Methylene bis(thiocyanate)
16266	9006-42-2	Metiram-zinc
16267	51218-45-2	Metolachlor
16268	21087-64-9	Metribuzin
16269	74223-64-6	Metsulfuron-methyl
16270	7786-34-7	Mevinphos
16271	113-48-4	MGK
16272	51596-11-3	Milbemectin
16273	2212-67-1	Molinate
16274	131-70-4	Monobutyl phthalate
16275	6923-22-4	Monocrotophos
16276	88671-89-0	Myclobutanil
16277	300-76-5	Naled
16278	15299-99-7	Napropamide
16279	50-65-7	Niclosamide
16280	1929-82-4	Nitrapyrin
16281	27314-13-2	Norflurazon
16282	116714-46-6	Novaluron
16283	19044-88-3	Oryzalin
16284	19666-30-9	Oxadiazon

(continued)

GCID	CASRN	Name
16285	23135-22-0	Oxamyl
16286	144651-06-9	Oxasulfuron
16287	42874-03-3	Oxyfluorfen
16288	6153-64-6	Oxytetracycline dihydrate
16289	76738-62-0	Paclobutrazol
16290	56-38-2	Parathion
16291	298-00-0	Parathion-methyl
16292	40487-42-1	Pendimethalin
16293	219714-96-2	Penoxsulam
16294	1763-23-1	Perfluorooctane sulfonic acid
16295	335-67-1	Perfluorooctanoic acid
16296	52645-53-1	Permethrin
16297	122-99-6	Phenoxyethanol
16298	2310-17-0	Phosalone
16161	4376-20-9	Phthalic acid, mono-2-ethylhexyl ester
16299	1918-02-1	Picloram
16300	51-03-6	Piperonyl butoxide
16301	23103-98-2	Pirimicarb
16302	29232-93-7	Pirimiphos-methyl
16303	23031-36-9	Prallethrin
16304	86209-51-0	Primisulfuron-methyl
16305	67747-09-5	Prochloraz
16306	29091-21-2	Prodiamine
16307	41198-08-7	Profenofos
16308	127277-53-6	Prohexadione-calcium
16309	1610-18-0	Prometon
16310	7287-19-6	Prometryn
16311	25606-41-1	Propamocarb hydrochloride
16312	709-98-8	Propanil
16313	2312-35-8	Propargite
16314	139-40-2	Propazine
16315	31218-83-4	Propetamphos
16316	60207-90-1	Propiconazole

(continued)

GCID	CASRN	Name
16317	114-26-1	Propoxur
16318	181274-15-7	Propoxycarbazone-sodium
16319	23950-58-5	Propyzamide
16320	94125-34-5	Prosulfuron
16321	123312-89-0	Pymetrozine
16322	175013-18-0	Pyraclostrobin
16323	129630-19-9	Pyraflufen-ethyl
16324	96489-71-3	Pyridaben
16325	53112-28-0	Pyrimethanil
16326	95737-68-1	Pyriproxyfen
16327	123343-16-8	Pyrithiobac-sodium
16328	84087-01-4	Quinclorac
16329	124495-18-7	Quinoxyfen
16330	82-68-8	Quintozene
16331	76578-14-8	Quizalofop-ethyl
16332	10453-86-8	Resmethrin
16333	122931-48-0	Rimsulfuron
16334	83-79-4	Rotenone
16176	28434-00-6	S-Bioallethrin
16335	74051-80-2	Sethoxydim
16336	122-34-9	Simazine
16337	148477-71-8	Spirodiclofen
16338	118134-30-8	Spiroxamine
16339	122836-35-5	Sulfentrazone
16340	87-90-1	Symclosene
16341	21564-17-0	TCMTB
16342	112410-23-8	Tebufenozide
16343	119168-77-3	Tebufenpyrad
16344	96182-53-5	Tebupirimfos
16345	34014-18-1	Tebuthiuron
16346	79538-32-2	Tefluthrin
16347	149979-41-9	Tepraloxydim
16348	5902-51-2	Terbacil

(continued)

GCID	CASRN	Name
16349	112281-77-3	Tetraconazole
16350	7696-12-0	Tetramethrin
16351	148-79-8	Thiabendazole
16352	111988-49-9	Thiacloprid
16353	153719-23-4	Thiamethoxam
16354	117718-60-2	Thiazopyr
16355	51707-55-2	Thidiazuron
16356	28249-77-6	Thiobencarb
16357	59669-26-0	Thiodicarb
16358	23564-05-8	Thiophanate-methyl
16359	137-26-8	Thiram
16360	87820-88-0	Tralkoxydim
16361	43121-43-3	Triadimefon
16362	55219-65-3	Triadimenol
16363	2303-17-5	Tri-allate
16364	82097-50-5	Triasulfuron
16365	101200-48-0	Tribenuron-methyl
16366	78-48-8	Tribufos
16367	52-68-6	Trichlorfon
16368	55335-06-3	Triclopyr
16369	3380-34-5	Triclosan
16370	141517-21-7	Trifloxystrobin
16371	199119-58-9	Trifloxysulfuron-sodium
16372	68694-11-1	Triflumizole
16373	1582-09-8	Trifluralin
16374	126535-15-7	Triflusulfuron-methyl
16375	131983-72-7	Triticonazole
16376	50471-44-8	Vinclozolin
16377	156052-68-5	Zoxamide

A.2 Experimental Results

Table A.2: Area under the ROC curve of multi-label classification and filtering on the ToxCast data set with classifiers Ensemble of Classifier Chains and multi-label classification using Boolean matrix decomposition. In the label names, C is short for CHR, D for DEV, and M for MGR. Mo is an abbreviation of Mouse, Rt abbreviates Rat, and Rabbit is abbreviated Rb, developmental is shortened to dev, maternal to mat. Due to the diverse quality of the labels in terms of class distribution and number of missing values, the table might seem unusual. The dashes indicate that no AUC value could be calculated. This is for example the case when in the calculation of the AUC a division by 0 occurs, which can be the case when there are no positive or no negative predictions. As the labels are very sparse, this occurs frequently. Naturally, in the case of the filtered classifier, this occurs more often, as many labels are never predicted, others only in a few folds. The sparsity of the data set also results in some cases where the AUC is 0.0 or 1.0. This can happen if there are only few target values which are predicted all correctly (hence resulting in an AUC of 1.0), or incorrectly (in the case of an AUC of 0.0). In the majority of the cases, these labels have less than 5 predicted instances for that setting, sometimes only one instance is actually predicted.

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Mo_AdrenalGland_1_AnyLesion	0.459	0.018	-	0.500	-
C_Mo_AdrenalGland_2_PreneoplasticLesion	-	0.325	-	0.416	-
C_Mo_AdrenalGland_3_NeoplasticLesion	0.422	0.353	0.188	0.466	-
C_Mo_ArteryGeneral_1_AnyLesion	-	0.476	-	0.513	-
C_Mo_ArteryGeneral_2_PreneoplasticLesion	0.388	0.000	0.626	0.434	0.463
C_Mo_ArteryGeneral_3_NeoplasticLesion	-	0.548	-	0.509	-
C_Mo_Blood_1_AnyLesion	0.339	0.435	0.152	0.522	-
C_Mo_Blood_2_PreneoplasticLesion	-	0.436	-	0.475	-
C_Mo_Blood_3_NeoplasticLesion	0.612	0.040	-	0.381	-
C_Mo_Bloodvessel_1_AnyLesion	0.492	0.156	-	0.455	-
C_Mo_Bloodvessel_2_PreneoplasticLesion	0.436	0.442	-	0.473	-
C_Mo_Bloodvessel_3_NeoplasticLesion	-	0.425	-	0.462	-
C_Mo_BoneMarrow_1_AnyLesion	0.444	0.525	-	0.533	-
C_Mo_BoneMarrow_2_PreneoplasticLesion	0.543	0.305	0.875	0.499	0.527
C_Mo_BoneMarrow_3_NeoplasticLesion	0.474	-	1.000	-	-
C_Mo_Bone_1_AnyLesion	-	0.508	-	0.538	-
C_Mo_Bone_2_PreneoplasticLesion	0.618	0.408	-	0.498	-
C_Mo_Bone_3_NeoplasticLesion	-	-	-	-	-
C_Mo_Brain_1_AnyLesion	-	0.337	-	0.495	-
C_Mo_Brain_2_PreneoplasticLesion	0.397	-	-	-	-
C_Mo_Brain_3_NeoplasticLesion	-	0.499	-	0.586	-
C_Mo_ClitoralGland_1_AnyLesion	0.684	0.043	0.516	0.381	-
C_Mo_ClitoralGland_2_PreneoplasticLesion	0.480	0.008	-	0.492	-
C_Mo_ClitoralGland_3_NeoplasticLesion	-	0.370	-	0.476	-
C_Mo_CoagulatingGland_1_AnyLesion	0.579	0.479	0.439	0.453	0.458
C_Mo_CoagulatingGland_2_PreneoplasticLesion	-	-	-	-	-
C_Mo_CoagulatingGland_3_NeoplasticLesion	0.515	0.361	-	0.439	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Mo_CoagulatingGland_3_NeoplasticLesion	0.580	0.510	-	0.574	-
C_Mo_Ear_1_AnyLesion	-	-	-	-	-
C_Mo_Ear_2_PreneoplasticLesion	0.543	0.501	1.000	0.462	-
C_Mo_Ear_3_NeoplasticLesion	-	0.264	-	0.500	-
C_Mo_Epididymis_1_AnyLesion	0.693	0.418	-	0.500	-
C_Mo_Epididymis_2_PreneoplasticLesion	0.435	0.433	-	0.502	-
C_Mo_Epididymis_3_NeoplasticLesion	0.594	0.000	0.718	0.434	0.250
C_Mo_Esophagus_1_AnyLesion	0.417	-	0.201	-	-
C_Mo_Esophagus_2_PreneoplasticLesion	-	0.541	-	0.546	-
C_Mo_Esophagus_3_NeoplasticLesion	0.438	0.015	-	0.500	-
C_Mo_Eye_1_AnyLesion	0.468	-	0.000	-	-
C_Mo_Eye_2_PreneoplasticLesion	0.476	0.231	-	0.512	-
C_Mo_Eye_3_NeoplasticLesion	-	0.015	-	0.494	-
C_Mo_Gallbladder_1_AnyLesion	0.714	-	-	-	-
C_Mo_Gallbladder_2_PreneoplasticLesion	0.580	0.000	-	0.434	0.000
C_Mo_Gallbladder_3_NeoplasticLesion	-	0.487	-	0.450	-
C_Mo_HarderianGland_1_AnyLesion	0.551	0.424	0.600	0.365	-
C_Mo_HarderianGland_2_PreneoplasticLesion	0.508	-	-	-	-
C_Mo_HarderianGland_3_NeoplasticLesion	0.438	0.482	-	0.453	-
C_Mo_Heart_1_AnyLesion	0.459	-	-	-	-
C_Mo_Heart_2_PreneoplasticLesion	0.506	-	-	-	-
C_Mo_Heart_3_NeoplasticLesion	0.502	0.308	-	0.500	-
C_Mo_IntestineLarge_1_AnyLesion	0.472	0.276	-	0.417	-
C_Mo_IntestineLarge_2_PreneoplasticLesion	0.506	0.281	0.515	0.444	-
C_Mo_IntestineLarge_3_NeoplasticLesion	0.415	-	0.000	-	0.600
C_Mo_IntestineSmall_1_AnyLesion	0.439	0.431	0.166	0.304	-
C_Mo_IntestineSmall_2_PreneoplasticLesion	0.538	0.521	-	0.489	-
C_Mo_IntestineSmall_3_NeoplasticLesion	0.476	0.017	-	0.494	0.303
C_Mo_KidneyPathology	0.519	-	0.416	-	-
C_Mo_Kidney_1_AnyLesion	0.457	-	0.181	-	-
C_Mo_Kidney_2_PreneoplasticLesion	0.431	0.283	-	0.583	-
C_Mo_Kidney_3_NeoplasticLesion	0.469	0.461	-	0.405	-
C_Mo_LacrimalGland_1_AnyLesion	0.525	0.567	-	0.534	-
C_Mo_LacrimalGland_2_PreneoplasticLesion	0.477	0.474	0.666	0.579	1.000
C_Mo_LacrimalGland_3_NeoplasticLesion	-	0.467	-	0.362	-
C_Mo_Larynx_1_AnyLesion	0.687	-	0.287	-	-
C_Mo_Larynx_2_PreneoplasticLesion	0.607	-	0.216	-	-
C_Mo_Larynx_3_NeoplasticLesion	-	0.060	-	0.377	-
C_Mo_LiverHypertrophy	0.480	-	-	-	-
C_Mo_LiverNecrosis	-	0.556	-	0.489	-
C_Mo_LiverProlifeRtiveLesions	0.529	0.310	-	0.459	-
C_Mo_LiverTumors	0.427	0.453	-	0.433	-
C_Mo_Liver_1_AnyLesion	0.482	0.038	-	0.496	0.750
C_Mo_Liver_2_PreneoplasticLesion	-	0.178	-	0.498	-
C_Mo_Liver_3_NeoplasticLesion	-	-	-	-	-
C_Mo_LungTumors	-	-	-	-	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Mo_Lung_1_AnyLesion	0.444	-	0.384	-	0.875
C_Mo_Lung_2_PreneoplasticLesion	0.463	-	-	-	-
C_Mo_Lung_3_NeoplasticLesion	-	0.404	-	0.665	-
C_Mo_LymphNode_1_AnyLesion	0.516	0.040	-	0.470	0.277
C_Mo_LymphNode_2_PreneoplasticLesion	0.548	0.416	-	0.500	-
C_Mo_LymphNode_3_NeoplasticLesion	0.462	-	-	-	0.500
C_Mo_MammaryGland_1_AnyLesion	0.492	0.398	-	0.548	0.166
C_Mo_MammaryGland_2_PreneoplasticLesion	0.415	0.313	0.531	0.512	0.900
C_Mo_MammaryGland_3_NeoplasticLesion	0.534	0.111	0.132	0.468	-
C_Mo_Mesentery_1_AnyLesion	0.457	-	0.623	-	-
C_Mo_Mesentery_2_PreneoplasticLesion	0.449	0.558	-	0.498	-
C_Mo_Mesentery_3_NeoplasticLesion	0.435	0.492	-	0.529	-
C_Mo_Nerve_1_AnyLesion	0.473	-	-	-	-
C_Mo_Nerve_2_PreneoplasticLesion	0.759	0.456	-	0.478	-
C_Mo_Nerve_3_NeoplasticLesion	0.456	-	-	-	-
C_Mo_Nose_1_AnyLesion	-	0.021	-	0.500	-
C_Mo_Nose_2_PreneoplasticLesion	0.569	0.555	0.280	0.480	0.764
C_Mo_Nose_3_NeoplasticLesion	-	0.267	-	0.558	-
C_Mo_OralMucosa_1_AnyLesion	-	0.010	-	0.380	-
C_Mo_OralMucosa_2_PreneoplasticLesion	0.484	0.504	-	0.558	-
C_Mo_OralMucosa_3_NeoplasticLesion	0.516	0.554	-	0.492	-
C_Mo_Ovary_1_AnyLesion	0.539	0.220	0.491	0.469	-
C_Mo_Ovary_2_PreneoplasticLesion	0.608	0.544	0.525	0.474	0.500
C_Mo_Ovary_3_NeoplasticLesion	0.657	0.228	0.948	0.596	-
C_Mo_PaRthyroidGland_1_AnyLesion	0.468	0.013	-	0.494	-
C_Mo_PaRthyroidGland_2_PreneoplasticLesion	0.696	0.517	0.555	0.586	-
C_Mo_PaRthyroidGland_3_NeoplasticLesion	0.492	0.304	0.703	0.428	-
C_Mo_Pancreas_1_AnyLesion	-	-	-	-	-
C_Mo_Pancreas_2_PreneoplasticLesion	-	0.351	-	0.521	-
C_Mo_Pancreas_3_NeoplasticLesion	-	0.423	-	0.507	-
C_Mo_Penis_1_AnyLesion	0.396	0.146	0.483	0.490	-
C_Mo_Penis_2_PreneoplasticLesion	-	-	-	-	-
C_Mo_Penis_3_NeoplasticLesion	-	0.324	-	0.436	-
C_Mo_Peritoneum_1_AnyLesion	-	0.504	-	0.446	-
C_Mo_Peritoneum_2_PreneoplasticLesion	-	0.306	-	0.494	-
C_Mo_Peritoneum_3_NeoplasticLesion	0.562	-	-	-	-
C_Mo_PituitaryGland_1_AnyLesion	-	0.496	-	0.399	-
C_Mo_PituitaryGland_2_PreneoplasticLesion	0.524	0.184	-	0.483	0.500
C_Mo_PituitaryGland_3_NeoplasticLesion	0.515	0.287	-	0.462	-
C_Mo_PreputialGland_1_AnyLesion	-	0.235	-	0.453	-
C_Mo_PreputialGland_2_PreneoplasticLesion	0.612	0.615	0.452	0.466	0.428
C_Mo_PreputialGland_3_NeoplasticLesion	0.480	0.403	-	0.550	-
C_Mo_Prostate_1_AnyLesion	0.948	0.614	-	0.457	-
C_Mo_Prostate_2_PreneoplasticLesion	0.415	0.127	-	0.606	-
C_Mo_Prostate_3_NeoplasticLesion	0.528	-	0.478	-	-
C_Mo_Salivaryglands_1_AnyLesion	-	0.432	-	0.492	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Mo_Salivaryglands_2_PreneoplasticLesion	0.480	0.376	-	0.487	-
C_Mo_Salivaryglands_3_NeoplasticLesion	-	-	-	-	-
C_Mo_SeminalVesicle_1_AnyLesion	0.765	-	-	-	-
C_Mo_SeminalVesicle_2_PreneoplasticLesion	0.376	0.487	-	0.338	-
C_Mo_SeminalVesicle_3_NeoplasticLesion	0.509	0.503	-	0.474	0.416
C_Mo_SkeletalMuscle_1_AnyLesion	0.455	-	-	-	-
C_Mo_SkeletalMuscle_2_PreneoplasticLesion	0.489	-	-	-	-
C_Mo_SkeletalMuscle_3_NeoplasticLesion	0.459	0.026	-	0.500	-
C_Mo_Skin_1_AnyLesion	0.457	0.400	-	0.475	-
C_Mo_Skin_2_PreneoplasticLesion	0.563	0.246	0.807	0.355	-
C_Mo_Skin_3_NeoplasticLesion	0.474	0.440	-	0.460	-
C_Mo_Spinalcord_1_AnyLesion	-	0.533	-	0.489	-
C_Mo_Spinalcord_2_PreneoplasticLesion	-	0.303	-	0.564	-
C_Mo_Spinalcord_3_NeoplasticLesion	0.545	0.454	-	0.430	0.166
C_Mo_Spleen_1_AnyLesion	0.646	0.036	-	0.5	-
C_Mo_Spleen_2_PreneoplasticLesion	0.478	0.449	-	0.461	-
C_Mo_Spleen_3_NeoplasticLesion	0.576	0.494	-	0.447	-
C_Mo_Stomach_1_AnyLesion	0.638	0.014	-	0.337	-
C_Mo_Stomach_2_PreneoplasticLesion	-	-	-	-	-
C_Mo_Stomach_3_NeoplasticLesion	0.468	0.490	0.644	0.472	-
C_Mo_Testes_1_AnyLesion	0.301	0.458	-	0.548	-
C_Mo_Testes_2_PreneoplasticLesion	-	0.047	-	0.500	-
C_Mo_Testes_3_NeoplasticLesion	-	0.064	-	0.468	-
C_Mo_Thymus_1_AnyLesion	0.461	-	-	-	-
C_Mo_Thymus_2_PreneoplasticLesion	0.463	0.443	-	0.486	-
C_Mo_Thymus_3_NeoplasticLesion	0.539	0.147	0.480	0.500	-
C_Mo_ThyroidGland_1_AnyLesion	0.541	0.292	0.166	0.491	0.666
C_Mo_ThyroidGland_2_PreneoplasticLesion	-	0.506	-	0.532	-
C_Mo_ThyroidGland_3_NeoplasticLesion	0.908	0.169	-	0.498	0.100
C_Mo_TissueNOS_1_AnyLesion	-	0.531	-	0.435	-
C_Mo_TissueNOS_2_PreneoplasticLesion	-	0.468	-	0.451	-
C_Mo_TissueNOS_3_NeoplasticLesion	-	-	-	-	-
C_Mo_Tongue_1_AnyLesion	0.483	0.109	0.700	0.498	-
C_Mo_Tongue_2_PreneoplasticLesion	-	-	-	-	-
C_Mo_Tongue_3_NeoplasticLesion	0.661	-	-	-	0.416
C_Mo_Tooth_1_AnyLesion	-	0.520	-	0.576	-
C_Mo_Tooth_2_PreneoplasticLesion	0.544	0.017	0.875	0.498	-
C_Mo_Tooth_3_NeoplasticLesion	0.529	0.476	-	0.500	-
C_Mo_Trachea_1_AnyLesion	-	0.460	-	0.507	-
C_Mo_Trachea_2_PreneoplasticLesion	0.645	0.395	-	0.609	1.000
C_Mo_Trachea_3_NeoplasticLesion	0.604	0.122	-	0.486	-
C_Mo_Tumorigen	0.474	0.413	-	0.365	-
C_Mo_UncertainPrimarySite_1_AnyLesion	0.475	0.392	-	0.432	-
C_Mo_UncertainPrimarySite_2_PreneoplasticLesion	0.487	0.209	-	0.500	0.166
C_Mo_UncertainPrimarySite_3_NeoplasticLesion	0.532	0.453	-	0.510	-
C_Mo_Ureter_1_AnyLesion	0.537	0.395	-	0.544	0.500

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Mo_Ureter_2_PreneoplasticLesion	0.554	-	-	-	0.250
C_Mo_Ureter_3_NeoplasticLesion	0.650	-	-	-	-
C_Mo_Urethra_1_AnyLesion	-	0.476	-	0.523	-
C_Mo_Urethra_2_PreneoplasticLesion	-	-	-	-	-
C_Mo_Urethra_3_NeoplasticLesion	0.467	0.085	-	0.494	-
C_Mo_UrinaryBladder_1_AnyLesion	0.720	0.279	-	0.400	-
C_Mo_UrinaryBladder_2_PreneoplasticLesion	0.396	0.032	0.537	0.137	0.500
C_Mo_UrinaryBladder_3_NeoplasticLesion	-	0.209	-	0.673	-
C_Mo_Uterus_1_AnyLesion	0.558	0.461	-	0.453	-
C_Mo_Uterus_2_PreneoplasticLesion	0.591	-	0.500	-	-
C_Mo_Uterus_3_NeoplasticLesion	0.614	0.462	0.777	0.488	-
C_Mo_Vagina_1_AnyLesion	0.553	-	0.709	-	-
C_Mo_Vagina_2_PreneoplasticLesion	0.480	0.587	-	0.431	-
C_Mo_Vagina_3_NeoplasticLesion	0.546	0.034	-	0.468	0.166
C_Mo_ZymbalsGland_1_AnyLesion	-	-	-	-	-
C_Mo_ZymbalsGland_2_PreneoplasticLesion	-	0.200	-	0.456	-
C_Mo_ZymbalsGland_3_NeoplasticLesion	0.473	0.008	-	0.500	-
C_Rt_AdrenalGland_1_AnyLesion	0.497	0.406	1.000	0.494	-
C_Rt_AdrenalGland_2_PreneoplasticLesion	-	-	-	-	-
C_Rt_AdrenalGland_3_NeoplasticLesion	-	0.065	-	0.449	-
C_Rt_ArteryGeneral_1_AnyLesion	0.567	0.213	0.471	0.545	-
C_Rt_ArteryGeneral_2_PreneoplasticLesion	-	0.253	-	0.536	-
C_Rt_ArteryGeneral_3_NeoplasticLesion	0.650	0.527	1.000	0.424	-
C_Rt_Blood_1_AnyLesion	-	-	-	-	-
C_Rt_Blood_2_PreneoplasticLesion	0.451	0.342	-	0.478	-
C_Rt_Blood_3_NeoplasticLesion	0.469	-	-	-	-
C_Rt_Bloodvessel_1_AnyLesion	-	0.353	-	0.472	-
C_Rt_Bloodvessel_2_PreneoplasticLesion	0.841	0.428	-	0.339	-
C_Rt_Bloodvessel_3_NeoplasticLesion	0.346	-	-	-	0.857
C_Rt_BoneMarrow_1_AnyLesion	0.562	0.004	0.456	0.500	0.416
C_Rt_BoneMarrow_2_PreneoplasticLesion	-	0.457	-	0.485	-
C_Rt_BoneMarrow_3_NeoplasticLesion	0.526	-	0.285	-	-
C_Rt_Bone_1_AnyLesion	0.415	0.512	-	0.443	-
C_Rt_Bone_2_PreneoplasticLesion	-	0.143	-	0.500	-
C_Rt_Bone_3_NeoplasticLesion	-	0.447	-	0.558	-
C_Rt_Brain_1_AnyLesion	-	-	-	-	-
C_Rt_Brain_2_PreneoplasticLesion	0.405	0.551	-	0.513	0.833
C_Rt_Brain_3_NeoplasticLesion	0.632	-	-	-	-
C_Rt_CholinesteraseInhibition	0.378	0.356	0.368	0.494	0.692
C_Rt_ClitoralGland_1_AnyLesion	0.518	0.581	0.300	0.437	-
C_Rt_ClitoralGland_2_PreneoplasticLesion	-	-	-	-	-
C_Rt_ClitoralGland_3_NeoplasticLesion	0.537	-	1.000	-	-
C_Rt_CoagulatingGland_1_AnyLesion	0.371	0.529	-	0.479	-
C_Rt_CoagulatingGland_2_PreneoplasticLesion	0.479	-	-	-	-
C_Rt_CoagulatingGland_3_NeoplasticLesion	0.564	0.478	0.900	0.495	-
C_Rt_Ear_1_AnyLesion	0.636	-	-	-	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Rt_Ear_2_PreneoplasticLesion	0.539	-	0.600	-	0.406
C_Rt_Ear_3_NeoplasticLesion	-	-	-	-	-
C_Rt_Epididymis_1_AnyLesion	0.512	-	0.333	-	-
C_Rt_Epididymis_2_PreneoplasticLesion	-	0.438	-	0.418	-
C_Rt_Epididymis_3_NeoplasticLesion	0.566	-	-	-	-
C_Rt_Esophagus_1_AnyLesion	0.546	0.140	-	0.498	0.500
C_Rt_Esophagus_2_PreneoplasticLesion	0.457	0.371	-	0.632	-
C_Rt_Esophagus_3_NeoplasticLesion	-	0.514	-	0.460	-
C_Rt_Eye_1_AnyLesion	-	-	-	-	-
C_Rt_Eye_2_PreneoplasticLesion	0.459	0.155	-	0.496	-
C_Rt_Eye_3_NeoplasticLesion	0.510	0.036	-	0.462	-
C_Rt_Gallbladder_1_AnyLesion	0.534	0.021	-	0.500	-
C_Rt_Gallbladder_2_PreneoplasticLesion	0.416	0.006	-	0.500	-
C_Rt_Gallbladder_3_NeoplasticLesion	0.566	-	1.000	-	0.833
C_Rt_HarderianGland_1_AnyLesion	0.474	-	-	-	-
C_Rt_HarderianGland_2_PreneoplasticLesion	0.495	0.532	0.602	0.586	0.583
C_Rt_HarderianGland_3_NeoplasticLesion	0.449	0.383	-	0.426	-
C_Rt_Heart_1_AnyLesion	-	0.507	-	0.453	-
C_Rt_Heart_2_PreneoplasticLesion	0.639	0.511	0.600	0.339	-
C_Rt_Heart_3_NeoplasticLesion	0.440	0.643	-	0.444	-
C_Rt_IntestineLarge_1_AnyLesion	-	-	-	-	-
C_Rt_IntestineLarge_2_PreneoplasticLesion	0.459	0.578	-	0.548	-
C_Rt_IntestineLarge_3_NeoplasticLesion	0.510	0.535	0.666	0.616	-
C_Rt_IntestineSmall_1_AnyLesion	-	0.128	-	0.500	-
C_Rt_IntestineSmall_2_PreneoplasticLesion	0.454	-	0.900	-	0.500
C_Rt_IntestineSmall_3_NeoplasticLesion	0.501	0.016	0.530	0.500	-
C_Rt_KidneyNephropathy	0.497	-	0.888	-	-
C_Rt_KidneyProlifeRtiveLesions	-	-	-	-	-
C_Rt_Kidney_1_AnyLesion	0.519	0.514	0.621	0.505	-
C_Rt_Kidney_2_PreneoplasticLesion	0.481	-	0.150	-	-
C_Rt_Kidney_3_NeoplasticLesion	-	0.328	-	0.552	-
C_Rt_LacrimalGland_1_AnyLesion	0.560	-	-	-	-
C_Rt_LacrimalGland_2_PreneoplasticLesion	-	0.340	-	0.512	-
C_Rt_LacrimalGland_3_NeoplasticLesion	0.587	-	0.550	-	-
C_Rt_Larynx_1_AnyLesion	-	-	-	-	-
C_Rt_Larynx_2_PreneoplasticLesion	0.497	0.156	-	0.480	-
C_Rt_Larynx_3_NeoplasticLesion	0.434	0.437	-	0.536	-
C_Rt_LiverHypertrophy	0.499	0.000	0.456	0.434	-
C_Rt_LiverNecrosis	0.480	0.371	-	0.340	-
C_Rt_LiverProlifeRtiveLesions	0.979	-	-	-	-
C_Rt_LiverTumors	-	-	-	-	-
C_Rt_Liver_1_AnyLesion	0.445	-	-	-	-
C_Rt_Liver_2_PreneoplasticLesion	0.429	0.446	-	0.491	-
C_Rt_Liver_3_NeoplasticLesion	0.474	-	-	-	-
C_Rt_Lung_1_AnyLesion	0.476	0.361	-	0.426	-
C_Rt_Lung_2_PreneoplasticLesion	0.676	0.432	-	0.452	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Rt_Lung_3_NeoplasticLesion	0.562	0.432	-	0.529	0.470
C_Rt_LymphNode_1_AnyLesion	-	0.555	-	0.568	-
C_Rt_LymphNode_2_PreneoplasticLesion	0.492	0.036	-	0.500	0.273
C_Rt_LymphNode_3_NeoplasticLesion	0.483	0.452	-	0.498	-
C_Rt_MammaryGland_1_AnyLesion	0.653	-	-	-	-
C_Rt_MammaryGland_2_PreneoplasticLesion	0.551	0.535	0.295	0.484	0.250
C_Rt_MammaryGland_3_NeoplasticLesion	-	0.302	-	0.483	-
C_Rt_Mesentery_1_AnyLesion	0.537	0.448	-	0.334	-
C_Rt_Mesentery_2_PreneoplasticLesion	0.578	0.418	0.682	0.452	1.000
C_Rt_Mesentery_3_NeoplasticLesion	0.480	0.523	-	0.449	-
C_Rt_Nerve_1_AnyLesion	0.474	0.457	-	0.434	-
C_Rt_Nerve_2_PreneoplasticLesion	0.465	0.182	0.251	0.534	-
C_Rt_Nerve_3_NeoplasticLesion	-	0.020	-	0.500	-
C_Rt_Nose_1_AnyLesion	0.474	0.463	-	0.485	-
C_Rt_Nose_2_PreneoplasticLesion	-	0.467	-	0.380	-
C_Rt_Nose_3_NeoplasticLesion	0.544	0.065	0.822	0.500	0.457
C_Rt_OralMucosa_1_AnyLesion	-	0.529	-	0.462	-
C_Rt_OralMucosa_2_PreneoplasticLesion	0.455	-	-	-	-
C_Rt_OralMucosa_3_NeoplasticLesion	0.402	-	-	-	-
C_Rt_Ovary_1_AnyLesion	-	-	-	-	-
C_Rt_Ovary_2_PreneoplasticLesion	0.471	0.613	-	0.474	-
C_Rt_Ovary_3_NeoplasticLesion	-	0.511	-	0.532	-
C_Rt_PaRthyroidGland_1_AnyLesion	0.398	-	-	-	0.000
C_Rt_PaRthyroidGland_2_PreneoplasticLesion	-	-	-	-	-
C_Rt_PaRthyroidGland_3_NeoplasticLesion	0.498	0.174	-	0.500	-
C_Rt_Pancreas_1_AnyLesion	0.476	-	-	-	-
C_Rt_Pancreas_2_PreneoplasticLesion	-	0.084	-	0.498	-
C_Rt_Pancreas_3_NeoplasticLesion	0.543	-	0.690	-	-
C_Rt_Penis_1_AnyLesion	-	0.497	-	0.512	-
C_Rt_Penis_2_PreneoplasticLesion	0.457	0.262	-	0.448	-
C_Rt_Penis_3_NeoplasticLesion	0.435	-	0.844	-	-
C_Rt_Peritoneum_1_AnyLesion	-	0.437	-	0.468	-
C_Rt_Peritoneum_2_PreneoplasticLesion	0.479	0.034	-	0.449	-
C_Rt_Peritoneum_3_NeoplasticLesion	0.973	0.393	0.462	0.399	-
C_Rt_PituitaryGland_1_AnyLesion	0.539	0.408	-	0.500	-
C_Rt_PituitaryGland_2_PreneoplasticLesion	0.609	0.023	0.583	0.381	-
C_Rt_PituitaryGland_3_NeoplasticLesion	-	0.321	-	0.520	-
C_Rt_PreputialGland_1_AnyLesion	0.528	-	-	-	-
C_Rt_PreputialGland_2_PreneoplasticLesion	0.720	0.504	-	0.535	-
C_Rt_PreputialGland_3_NeoplasticLesion	0.450	0.360	-	0.461	-
C_Rt_Prostate_1_AnyLesion	0.697	0.368	-	0.525	-
C_Rt_Prostate_2_PreneoplasticLesion	0.583	-	0.625	-	0.575
C_Rt_Prostate_3_NeoplasticLesion	0.690	-	-	-	-
C_Rt_Salivaryglands_1_AnyLesion	-	0.495	-	0.511	-
C_Rt_Salivaryglands_2_PreneoplasticLesion	0.526	0.014	0.666	0.140	-
C_Rt_Salivaryglands_3_NeoplasticLesion	-	0.468	-	0.397	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Rt_SeminalVesicle_1_AnyLesion	0.597	0.416	-	0.500	-
C_Rt_SeminalVesicle_2_PreneoplasticLesion	0.465	0.107	-	0.434	0.757
C_Rt_SeminalVesicle_3_NeoplasticLesion	0.481	0.301	-	0.496	-
C_Rt_SkeletalMuscle_1_AnyLesion	0.453	0.471	-	0.498	-
C_Rt_SkeletalMuscle_2_PreneoplasticLesion	0.589	-	-	-	-
C_Rt_SkeletalMuscle_3_NeoplasticLesion	-	-	-	-	-
C_Rt_Skin_1_AnyLesion	-	0.043	-	0.500	-
C_Rt_Skin_2_PreneoplasticLesion	0.519	0.410	0.214	0.437	0.540
C_Rt_Skin_3_NeoplasticLesion	0.429	0.493	-	0.449	-
C_Rt_Spinalcord_1_AnyLesion	0.435	0.023	-	0.381	-
C_Rt_Spinalcord_2_PreneoplasticLesion	0.627	0.616	0.416	0.425	-
C_Rt_Spinalcord_3_NeoplasticLesion	0.474	0.181	-	0.500	-
C_Rt_SpleenPathology	0.414	0.442	-	0.462	-
C_Rt_Spleen_1_AnyLesion	-	-	-	-	-
C_Rt_Spleen_2_PreneoplasticLesion	-	0.231	-	0.500	-
C_Rt_Spleen_3_NeoplasticLesion	0.483	0.259	-	0.476	-
C_Rt_Stomach_1_AnyLesion	0.628	0.083	-	0.494	-
C_Rt_Stomach_2_PreneoplasticLesion	-	0.000	-	0.434	-
C_Rt_Stomach_3_NeoplasticLesion	0.509	0.197	0.625	0.433	-
C_Rt_Testes_1_AnyLesion	0.563	0.004	-	0.321	-
C_Rt_Testes_2_PreneoplasticLesion	0.517	0.266	-	0.500	-
C_Rt_Testes_3_NeoplasticLesion	-	0.481	-	0.437	-
C_Rt_TesticulaRtrophy	0.576	-	0.363	-	-
C_Rt_TesticularTumors	0.534	0.441	-	0.568	-
C_Rt_Thymus_1_AnyLesion	0.505	0.558	-	0.437	-
C_Rt_Thymus_2_PreneoplasticLesion	0.432	0.480	-	0.445	-
C_Rt_Thymus_3_NeoplasticLesion	0.480	0.485	-	0.486	-
C_Rt_ThyroidGland_1_AnyLesion	0.596	0.004	-	0.498	-
C_Rt_ThyroidGland_2_PreneoplasticLesion	0.441	0.499	0.271	0.501	-
C_Rt_ThyroidGland_3_NeoplasticLesion	-	-	-	-	-
C_Rt_ThyroidHyperplasia	-	0.162	-	0.434	-
C_Rt_ThyroidProlifeRtiveLesions	-	0.301	-	0.484	-
C_Rt_ThyroidTumors	0.455	0.515	-	0.462	-
C_Rt_TissueNOS_1_AnyLesion	0.535	0.453	0.505	0.467	-
C_Rt_TissueNOS_2_PreneoplasticLesion	0.555	0.337	0.431	0.459	-
C_Rt_TissueNOS_3_NeoplasticLesion	0.605	-	0.583	-	-
C_Rt_Tongue_1_AnyLesion	0.528	0.541	0.682	0.457	0.738
C_Rt_Tongue_2_PreneoplasticLesion	0.428	0.472	0.714	0.391	-
C_Rt_Tongue_3_NeoplasticLesion	0.552	0.388	0.491	0.414	-
C_Rt_Tooth_1_AnyLesion	-	0.060	-	0.500	-
C_Rt_Tooth_2_PreneoplasticLesion	0.596	-	0.500	-	-
C_Rt_Tooth_3_NeoplasticLesion	-	0.388	-	0.419	-
C_Rt_Trachea_1_AnyLesion	0.430	0.504	-	0.488	0.945
C_Rt_Trachea_2_PreneoplasticLesion	0.452	0.414	0.625	0.505	-
C_Rt_Trachea_3_NeoplasticLesion	0.435	0.326	-	0.436	-
C_Rt_Tumorigen	-	0.505	-	0.390	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
C_Rt_UncertainPrimarySite_1_AnyLesion	-	0.028	-	0.439	-
C_Rt_UncertainPrimarySite_2_PreneoplasticLesion	0.383	-	-	-	-
C_Rt_UncertainPrimarySite_3_NeoplasticLesion	0.562	0.047	0.500	0.500	0.727
C_Rt_Ureter_1_AnyLesion	-	-	-	-	-
C_Rt_Ureter_2_PreneoplasticLesion	-	-	-	-	-
C_Rt_Ureter_3_NeoplasticLesion	0.471	0.359	0.125	0.476	0.500
C_Rt_Urethra_1_AnyLesion	-	-	-	-	-
C_Rt_Urethra_2_PreneoplasticLesion	0.390	-	1.000	-	-
C_Rt_Urethra_3_NeoplasticLesion	-	0.440	-	0.531	-
C_Rt_UrinaryBladder_1_AnyLesion	0.637	0.051	0.472	0.500	-
C_Rt_UrinaryBladder_2_PreneoplasticLesion	-	0.023	-	0.377	-
C_Rt_UrinaryBladder_3_NeoplasticLesion	0.565	-	-	-	0.628
C_Rt_Uterus_1_AnyLesion	0.475	-	0.750	-	0.687
C_Rt_Uterus_2_PreneoplasticLesion	0.424	0.403	-	0.488	-
C_Rt_Uterus_3_NeoplasticLesion	-	-	-	-	-
C_Rt_Vagina_1_AnyLesion	0.793	-	-	-	-
C_Rt_Vagina_2_PreneoplasticLesion	-	-	-	-	-
C_Rt_Vagina_3_NeoplasticLesion	0.583	0.087	-	0.500	1.000
C_Rt_ZymbalsGland_1_AnyLesion	-	-	-	-	-
C_Rt_ZymbalsGland_2_PreneoplasticLesion	-	0.505	-	0.437	-
C_Rt_ZymbalsGland_3_NeoplasticLesion	0.416	-	0.115	-	0.846
D_Rb_Dev	0.511	0.568	-	0.546	0.772
D_Rb_Dev_Cardiovascular	0.530	-	-	-	-
D_Rb_Dev_Cardiovascular_Heart	0.475	0.388	-	0.482	-
D_Rb_Dev_Cardiovascular_MajorVessels	-	-	-	-	-
D_Rb_Dev_GeneralFetal	0.598	0.548	1.000	0.446	-
D_Rb_Dev_GeneralFetal_FetalWeightReduction	-	-	-	-	-
D_Rb_Dev_GeneralFetal_GeneralFetalPathology	0.618	-	-	-	0.333
D_Rb_Dev_GeneralFetal_SexualDevLandmar	-	0.045	-	0.498	-
D_Rb_Dev_Neurosensory	-	0.492	-	0.554	-
D_Rb_Dev_Neurosensory_Brain	0.440	0.515	-	0.420	-
D_Rb_Dev_Neurosensory_Eye	-	-	-	-	-
D_Rb_Dev_Orofacial	0.583	-	-	-	-
D_Rb_Dev_Orofacial_CleftLipPalate	-	-	-	-	-
D_Rb_Dev_Orofacial_JawHyoid	0.470	0.537	-	0.547	-
D_Rb_Dev_PregnancyRelated	-	0.446	-	0.445	-
D_Rb_Dev_PregnancyRelated_EmbryoFetalLoss	0.480	0.352	-	0.416	-
D_Rb_Dev_Skeletal	-	-	-	-	-
D_Rb_Dev_Skeletal_Appendicular	0.711	0.415	0.651	0.531	0.722
D_Rb_Dev_Skeletal_Axial	0.476	0.000	-	0.500	-
D_Rb_Dev_Skeletal_Cranial	-	0.263	-	0.454	-
D_Rb_Dev_Skeletal_Unclassified	0.521	-	0.500	-	-
D_Rb_Dev_Trunk	0.527	0.173	-	0.500	-
D_Rb_Dev_Trunk_BodyWall	-	0.004	-	0.500	-
D_Rb_Dev_Trunk_SplanchnicViscera	0.474	0.360	-	0.384	-
D_Rb_Dev_Urogenital	0.433	-	0.600	-	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
D_Rb_Dev_Urogenital_Renal	0.661	0.497	-	0.454	-
D_Rb_Dev_Urogenital_Ureteric	-	-	-	-	-
D_Rb_Mat	0.446	0.371	-	0.484	-
D_Rb_Mat_GeneralMat	-	0.552	-	0.371	-
D_Rb_Mat_GeneralMat_Systemic	-	0.049	-	0.377	-
D_Rb_Mat_PregnancyRelated	0.561	0.295	-	0.338	-
D_Rb_Mat_PregnancyRelated_MatPregLoss	0.798	0.288	-	0.728	-
D_Rb_Prenatal_Loss	0.496	-	0.483	-	-
D_Rt_Dev	0.476	0.472	-	0.538	-
D_Rt_Dev_Cardiovascular	0.340	0.407	-	0.444	0.227
D_Rt_Dev_Cardiovascular_Heart	0.556	0.640	-	0.451	-
D_Rt_Dev_Cardiovascular_MajorVessels	0.457	-	0.937	-	-
D_Rt_Dev_GeneralFetal	0.607	0.297	-	0.492	-
D_Rt_Dev_GeneralFetal_FetalWeightReduction	0.488	-	0.157	-	-
D_Rt_Dev_GeneralFetal_GeneralFetalPathology	-	0.389	-	0.483	-
D_Rt_Dev_GeneralFetal_SexualDevLandmark	0.402	0.119	-	0.500	-
D_Rt_Dev_Neurosensory	0.446	0.423	-	0.558	1.000
D_Rt_Dev_Neurosensory_Brain	0.586	-	0.500	-	-
D_Rt_Dev_Neurosensory_Eye	0.520	0.469	0.666	0.486	-
D_Rt_Dev_Orofacial	0.545	0.002	-	0.498	-
D_Rt_Dev_Orofacial_CleftLipPalate	0.474	-	-	-	-
D_Rt_Dev_Orofacial_JawHyoid	-	0.227	-	0.500	-
D_Rt_Dev_PregnancyRelated	0.520	0.372	-	0.494	0.150
D_Rt_Dev_PregnancyRelated_EmbryoFetalLoss	0.432	0.498	-	0.425	0.214
D_Rt_Dev_Skeletal	0.492	-	-	-	-
D_Rt_Dev_Skeletal_Appendicular	-	-	-	-	-
D_Rt_Dev_Skeletal_Axial	0.800	-	-	-	-
D_Rt_Dev_Skeletal_Cranial	0.597	0.533	0.266	0.492	-
D_Rt_Dev_Skeletal_Unclassified	0.593	-	-	-	0.318
D_Rt_Dev_Trunk	0.509	-	-	-	-
D_Rt_Dev_Trunk_BodyWall	0.489	-	-	-	-
D_Rt_Dev_Trunk_SplanchnicViscera	0.548	0.484	0.250	0.532	1.000
D_Rt_Dev_Urogenital	0.478	0.522	-	0.449	-
D_Rt_Dev_Urogenital_Genital	-	0.421	-	0.487	-
D_Rt_Dev_Urogenital_Renal	0.612	0.306	0.750	0.487	0.287
D_Rt_Dev_Urogenital_Ureteric	-	0.446	-	0.419	-
D_Rt_Mat	0.615	0.370	-	0.271	-
D_Rt_Mat_GeneralMat	0.626	0.344	0.546	0.424	0.333
D_Rt_Mat_GeneralMat_Systemic	-	-	-	-	-
D_Rt_Mat_PregnancyRelated	0.422	0.408	0.303	0.566	0.151
D_Rt_Mat_PregnancyRelated_MatPregLoss	-	0.330	-	0.402	-
D_Rt_Prenatal_Loss	0.427	0.183	-	0.234	-
M_Rt_Adrenal	0.480	0.495	-	0.492	-
M_Rt_DevLandmark	0.359	-	-	-	0.750
M_Rt_EndocrineOrgan	-	0.032	-	0.434	-
M_Rt_Epididymis	0.459	-	-	-	-

(continued)

Label	BR	ECC	ECC Filter	MLC-BMaD	MLC-BMaD Filter
M_Rt_FemaleReproductiveTract	0.431	-	-	-	-
M_Rt_Fertility	0.457	0.409	-	0.425	-
M_Rt_GestationalInterval	0.624	-	0.338	-	-
M_Rt_Implantations	0.424	-	-	-	0.416
M_Rt_Kidney	0.546	-	0.372	-	0.000
M_Rt_LactationPND21	0.413	0.479	-	0.508	-
M_Rt_LitterSize	-	0.013	-	0.494	-
M_Rt_LiveBirthPND1	0.448	-	-	-	-
M_Rt_Liver	0.475	0.426	-	0.536	-
M_Rt_MaleReproductiveTract	0.603	0.268	0.673	0.498	-
M_Rt_Mating	0.467	0.420	-	0.419	-
M_Rt_Ovary	0.441	0.000	-	0.459	-
M_Rt_Pituitary	0.419	0.254	0.500	0.507	-
M_Rt_Prostate	-	-	-	-	-
M_Rt_ReproductiveOutcome	0.363	0.302	-	0.433	-
M_Rt_ReproductivePerformance	-	0.624	-	0.500	-
M_Rt_SexualDevLandmark	0.541	-	0.238	-	-
M_Rt_Spleen	-	0.109	-	0.481	-
M_Rt_Testis	0.463	0.575	-	0.500	-
M_Rt_Thyroid	0.694	0.308	0.648	0.564	-
M_Rt_Uterus	0.467	0.359	-	0.579	-
M_Rt_ViabilityPND4	0.440	0.387	-	0.406	-

Index