Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

DOKTORSKA DISERTACIJA

Učenje pravil za napovedno razvrščanje

Bernard Ženko

Mentor: akad. prof. dr. Ivan Bratko
Somentor: prof. dr. Sašo Džeroski

Ljubljana, 2007

University of Ljubljana
Faculty of computer and information science

PHD THESIS

Learning predictive clustering rules

Bernard Ženko

Thesis supervisor: Acad. Prof. Dr. Ivan Bratko
Thesis co-supervisor: Prof. Dr. Sašo Džeroski

Ljubljana, 2007

A theory is something nobody believes, except the person who made it.
An experiment is something everybody believes, except the person who made it.

A. Einstein

Acknowledgements

I wish to thank my co-supervisor Prof. Dr. Sašo Džeroski for accepting me in his research group, introducing me to the field of machine learning, and patiently supervising my work. His ideas and support were crucial in the process of writing this dissertation.

I am grateful to my supervisor Acad. Prof. Dr. Ivan Bratko for his insightful comments and suggestions that significantly improved this dissertation.

I am especially grateful to Ljupčo Todorovski for numerous discussions, encouragement, and support. His insightful ideas and suggestions strongly influenced the work presented in this dissertation.

Great thanks goes to Jan Struyf for kindly providing *Clus,* an excellent data mining suite, within which predictive clustering rules were implemented. Thanks for the help and patience with countless questions I had regarding Clus.

Thanks to Damjan Demšar for resurrecting *FRS,* a system for learning regression rules, which was used in this thesis as a comparison method. Thanks to Martin Možina for providing *CN2-EVC,* a system for learning classification rules, which was also used as a comparison method. Thanks to Annalisa Appice, Luís Torgo, and Nada Lavrač for reading an earlier version of the manuscript and providing useful comments. Thanks to Ernest Ženko for proof-reading the extended abstract in Slovene.

Special thanks to Tea Tušar, Mili Bauer, Andraž Bežek, and other members of the Departments of Knowledge Technologies and Intelligent Systems at the Jožef Stefan Institute for providing a stimulative and friendly working environment.

Most of the work presented in this thesis was financially supported by the Slovenian Research Agency. I also wish to acknowledge the financial support of the IQ (Inductive Queries for Mining Patterns and Models) project funded by the European Commission.

At last, but not at least, I wish to thank my parents for their patience and unconditional support.

Contents

Abstract

The field of machine learning is concerned with methods that can automatically learn from experience. The experience is usually given in the form of learning examples from which machine learning methods can automatically learn a model. Sometimes it is very important that the learned model can be read by human, and one of the most understandable types of models are rule sets. The methods for learning models in the form of rule sets are the topic of this thesis.

The predictive clustering approach to rule learning presented here is based on ideas from two machine learning subareas, predictive modeling and clustering. Both areas are usually regarded as completely different tasks. Predictive modeling is concerned with the construction of models that can be used to predict some object's target property from the description of this object. Clustering, on the other hand, is concerned with grouping of objects into classes of similar objects, called clusters; there is no target property to be predicted, and usually no symbolic description of discovered clusters. However, predictive modeling methods that partition the example space, such as decision trees and rules are very similar to clustering. They partition the set of examples into subsets in which examples have similar values of the target variable, while clustering produces subsets in which examples have similar values of all descriptive variables. Predictive clustering approach builds on this similarity. It constructs clusters of examples that are similar to each other, but in general takes both the descriptive and the target variables into account, and associates a predictive model to each constructed cluster. Methods for predictive clustering enable us to construct models for predicting multiple target variables, which are normally simpler and more comprehensible than the corresponding set of models, each predicting a single variable.

To this day, predictive clustering has been restricted to decision tree methods. The goal of this thesis is to extend predictive clustering approach to methods for learning rules, i.e., to develop a method that deals with rule learning and clustering in a uniform way. Of the existing rule learning methods, majority are based on the sequential covering algorithm, originally designed for learning ordered rule lists for binary classification domains. We have developed a generalized version of this algorithm that enables learning of ordered or unordered rules, on single or multiple target classification or regression domains.

The newly developed algorithm is empirically evaluated on several single and multiple target classification and regression problems. Performance of the new method compares favorably to existing methods. Comparison of single target and multiple target prediction models shows that multiple target models offer comparable performance and drastically lower complexity than the corresponding sets of single target models.

Chapter 1

Introduction

There are several forms of learning, ranging from 'learning by being told' to 'learning by discovery'. In the first case, the learner is explicitly told what is to be learned, while in the second case, the learner autonomously discovers new concepts from observations or by planning and performing experiments in the environment. Between these two extremes lies learning from examples, also called *inductive learning* (Bratko, 2001). Depending on the feedback the learner gets during the learning process, the learning can be *supervised* or *unsupervised.* In the former case, a tutor or domain expert gives the learner direct feedback about the appropriateness of its performance, while in the latter case, the feedback is absent (Langley, 1996).

Machine learning builds on concepts from the fields of artificial intelligence, statistics, information theory, and many others. It studies computer programs that automatically improve with experience (Mitchell, 1997). The most researched type of machine learning is inductive machine learning, where the experience is given in the form of learning examples. Supervised inductive machine learning, sometimes also called *predictive modeling,* assumes that each learning example includes some *target property,* and the goal is to learn a model that accurately predicts this property. Unsupervised inductive machine learning, on the other hand, assumes no such target property to be predicted. The thesis at hand is concerned with methods for inductive machine learning that result in models written in the form of rules.

The predictive clustering approach to rule learning presented in this thesis is based on ideas from both supervised and unsupervised machine learning. As already mentioned, predictive modeling (supervised learning) is concerned with the construction of models that can be used to predict some object's target property from the *description* of this object. A predictive model is learned from a set of learning objects or examples, where each example comprises a description and a target part. Many different languages can be used for the formulation of the description part, ranging from propositional to first order logic, though the first one, also called *attribute-value representation,* is by far the most commonly used. On the other hand, the target part is almost always considered to be a single variable. In machine learning this variable is called a *class.* In the simplest case, the class is a boolean variable, and each example can either be *positive* or *negative.* The task of learning to predict a boolean variable is also called *concept learning,* because it can be interpreted as the task of learning whether an example belongs to a specific concept or not. In a more general case, the class can have more than two possible values, i.e., it can be a nominal variable (with more than two possible values). Regardless of the number of possible values, the task of learning to predict the value of a nominal variable is called a *classification* task. Another possibility is that the class is a numeric variable; in this case we have a *regression* task. Predictive models that address these tasks can take many different forms that range from linear equations to logic programs. Two commonly used types of models are decision trees (Quinlan, 1993) and rules (Flach and Lavrač, 2003). Their benefit is that they are comprehensible and can easily be read and analyzed by a human. Unlike some representations, e.g., linear regression equations, that treat the entire example space simultaneously, trees and rules divide the space of examples into subspaces, and for each subspace provide a simple prediction or a predictive (sub)model.

The second line of research related to this thesis comes from the field of unsupervised learning, or more specifically, *clustering* (Kaufman and Rousseeuw, 1990). Clustering in general is concerned with grouping of objects into classes of similar objects, called clusters. Each object is described using some language, and, just like in predictive modeling, the attribute-value representation is most commonly used for this purpose. The most notable difference, when compared to predictive modeling, is that the examples in clustering have no target property to be predicted. The task is to find such clusters that the similarity of examples within individual clusters is high, and, at the same time, the similarity of examples from different clusters is low. The similarity of examples is measured using some metric defined over the object space. Usually, the objects are described in terms of attribute's values, and are considered to be points in a multi-dimensional metric space where distances between all points are uniquely determined. Each cluster of objects can also be represented by a prototypical object or a *prototype,* which can be defined as the point in the metric space with the lowest average distance to all objects in the cluster. This way, distances between clusters of objects are also defined. Commonly, the final result of clustering is just a set of object clusters without their symbolic descriptions. Each cluster can be, however, considered a separate concept, and symbolic descriptions can be added to already constructed clusters; this approach is called *conceptual clustering* (Michalski, 1980). Another approach that results in clusters with symbolic descriptions is clustering with *clustering trees* (Blockeel, 1998). Namely, a decision tree can be regarded as a hierarchy of clusters where each node is a cluster. Here, every cluster has a symbolic description formed by the conjunction of conditions on the path from the root of the tree to the given node.

Because of the differences mentioned above, predictive modeling and clustering are usually regarded as completely different machine learning techniques. On the other hand, there are also several similarities that clustering shares with some predictive modeling methods, most notably with methods that partition the example space, such as decision trees (Langley, 1996). Decision trees partition the set of examples into subsets in which examples have similar values of the target variable, while clustering produces subsets in which examples have similar values of all descriptive variables. Based on this similarity, the *predictive clustering* (Blockeel, 1998; Blockeel et al., 1998) approach has been developed. As is common in 'ordinary' clustering, predictive clustering constructs clusters of examples that are similar to each other, but in general taking both the descriptive and the target variables into account. In addition, a predictive model is associated with each cluster which describes the cluster, and, based on the values of the descriptive variables, predicts the values of the target variables. Methods for predictive clustering enable us to construct models for predicting multiple target variables which are normally simpler and more comprehensible than the corresponding set of models, each predicting a single variable. Because similarity estimation is done by taking into account all the variables, predictive clustering is especially suitable for the analysis of noisy domains and domains with missing values for the target variables (Blockeel, 1998). So far, this approach has been limited to the tree learning methods. Our goal is to extend predictive clustering with methods for learning rules, which offer an alternative description language. Such methods would extend the applicability of the predictive clustering approach.

1.1 Goals

Our main goal is to extend the predictive clustering approach to methods for learning rules, i.e., to develop methods for learning predictive clustering rules. There exist many different methods for rule learning (Fürnkranz, 1999), however, they are almost exclusively based on the covering algorithm (Michalski, 1969). The covering algorithm was originally designed for concept learning in a way that only rules describing the positive examples are learned, while examples not covered by any rule are assumed to be negative. The algorithm can be naturally extended to learn ordered rules for both positive and negative values of a single target variable. Many of the rule learning methods extend the covering approach towards multi-class classification domains (a single target variable with more than two possible values), somewhat fewer methods extend it towards unordered rule learning, and just a few methods extend it towards regression domains. To our knowledge, however, there is no single method that would enable the induction of rules in general, for multi-class classification and regression domains, and would produce ordered and unordered rules. There is also no rule learning method that can induce rules for predicting multiple target variables simultaneously. The extension of predictive clustering towards rules calls for such a general rule learning method, and the development of such a method became the main subgoal of this thesis. Such a method would not only extend the applicability of the predictive clustering approach, but would also extend the rule learning methods themselves. In addition, our goal is also to evaluate the performance of the developed methods on a range of tasks, and to compare them to existing methods.

1.2 Original contributions

The work presented in this thesis comprises several contributions to the area of machine learning. First, we have developed a new method for learning unordered single target classification rules. It is loosely based on the commonly used rule learning method CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), but uses a generalized *weighted covering* algorithm (Gamberger and Lavrač, 2002).

Second, the developed method is generalized for learning ordered or unordered rules, on single or multiple target classification or regression domains. It uses a search heuristic that takes into account several rule quality measures and is applicable to all the above mentioned types of domains.

The third contribution of the thesis is the extension of the predictive clustering approach to models in the form of rules. The newly developed method combines rule induction and clustering. The search heuristic takes into account the values of both, the target and the descriptive attributes. Different weighting of these two types of attributes enable us to traverse from the predictive modeling setting to the clustering setting. Larger weights of the target attributes will result in rules with better predictive accuracy, while larger weights of the descriptive attributes will result in rules that represent more compact clusters. We expect the extension of predictive clustering towards rules will propagate its usage in new domains where, besides predictive accuracy, other properties of models, such as simplicity of models or compactness of clusters, are also important.

The final contribution is an extensive empirical evaluation of the newly developed method on single target classification and regression problems, as well as multiple target classification and regression problems. The performance of the new method is compared to some existing methods. On multiple target problems, the performance of single target and multiple target prediction is compared. The results show that multiple target models are generally of comparable accuracy as single target models, however, the former are much smaller than the corresponding sets of single target models. In addition, the influence of some algorithm parameters, such as the weights of target and descriptive attributes, is investigated.

1.3 Organization of the thesis

The thesis is organized as follows. This introductory chapter presented the immediate context of the thesis topic. It specified the goals set at the beginning of the thesis research and presented its main original contributions.

The second chapter gives the broader context of the thesis work. It gives some background and a brief overview of the research related to the work presented in the thesis. It includes the areas of predictive modeling, clustering, predictive clustering, and rule learning.

Chapter 3 presents a selection of quality measures for rules and rule sets. These measures are an essential part of the rule learning algorithm (presented in Chapter 4). Quality measures for single rules are used in the rule search process as a part of the search heuristic. Rule set quality measures are used for rule set evaluation in the process of rule set construction as well as in the process of the final rule set evaluation.

Chapter 4 presents the main contribution of the thesis, the learning algorithm for predictive clustering rules. We start by formally defining the task of learning predictive clustering rules and then present the top level algorithm. The major parts of the algorithm are presented, each in a separate section.

Chapter 5 presents the empirical evaluation of the newly developed method on single and multiple target classification and regression problems.

Finally, the last chapter concludes with a summary of the thesis, its original contributions, and a discussion of some directions for further research.

Chapter 2

Background and related work

The work presented in this thesis is about learning predictive clustering rules, and as such it is related to several areas of machine learning. Predictive clustering is an approach that combines predictive modeling and clustering. Predictive modeling, clustering, and predictive clustering are briefly presented in this chapter. Rule learning can be viewed as a part of predictive modeling, but because this area is of special importance to the thesis, it is presented in a separate section. Each section briefly presents the main ideas of the area as well as research in the area related to the topic of this thesis.

2.1 Predictive modeling

Predictive modeling or supervised learning (Hastie et al., 2001) is one of the traditional machine learning areas. The task of predictive modeling is defined as follows. Given is a set of *inputs*, which are assumed to have some influence on an *output*. Given is also a set of *learning* (or *training*) examples, each of them linking a specific instantiation of the inputs to the corresponding values of the output. The goal is to use the learning examples to learn a predictive model that predicts the output from the inputs.

The task described above is of practical importance to many domains, ranging from medicine and social sciences to various areas of engineering. It can mostly be interpreted in two ways. The first interpretation is common in system's theory and is the same as stated above. The inputs are regarded as influences of the environment to the system and a description of the state of the system, while the outputs represent the response of the system. Learning examples are regarded as measurements of the input and output quantities under different conditions, i.e., the environment's influences, states, and responses of the system. The task is then to construct a model of the system that predicts the system's response, based on the influences of the environment and the internal state of the system. In the second interpretation, every learning example is a representation of an object. The inputs represent a description of an object, and the outputs are some object's properties that we are interested in. The constructed model should be able to predict the object's properties from its description. Both interpretations are, of course, equivalent, and which one we use depends on the problem domain and personal taste. Still, the latter is more common in machine learning, and is also used in this thesis.

Each learning example is regarded as a representation of an object. Various languages can be used for encoding examples, but the *attribute-value representation* is by far the most commonly used. Here, the examples are represented by a set of variables or *attributes*, and its values determine each example. Variables can be of different types, but mostly we only deal with nominal and numeric variables. This representation comprises two parts, the input part, here we will call it the *description* part, and the output part, which we will call the *target* part. In this context we are talking about description and target variables or attributes. A description part of any nontrivial domain needs to have more than one attribute, the target part, however, traditionally consists of only one attribute, which in machine learning is called a *class.* If the class is a nominal variable, we have a *classification* task, and if it is a numeric variable, we have a *regression* task. In case we have more than one target attribute, the task is called *multiple target prediction.*[1]

There exist many different types of predictive models and methods for their learning. The selection of a model type for a given problem domain depends on many factors, one of them being also the intended use of the model. If the model is to be used for prediction only, e.g., to diagnose new patients for a certain disease, it should be as accurate as possible, while the comprehensibility of the model is not important. However, if we would like to use the model to get some insight in the problem domain, e.g., to study what causes the disease, the model still has to be accurate, but also (and sometimes above all) comprehensible. Unfortunately, many types of models, which are often highly accurate, are also completely incomprehensible. Among the comprehensible types of models the decision trees (Quinlan, 1993) and rules (Flach and Lavrač, 2003) are, besides equations, very common. Unlike equations, that treat the entire example space simultaneously, decision trees and rules divide the space of examples into subspaces, and for each provide a predictive submodel. This submodel is a constant value or, rarely, a simple equation. Models in the form of rules are the topic of this thesis.

[1]The multiple target prediction task is sometimes called multi-objective or vector prediction.

2.2 Clustering

Clustering (Kaufman and Rousseeuw, 1990), also called cluster analysis or data segmentation, is an unsupervised learning (Hastie et al., 2001) approach. It is concerned with grouping of objects into classes of similar objects, called clusters. The task is to find such clusters that the similarity of objects within individual clusters is high, while the similarity of objects from different clusters is low. Central to clustering is the notion of a degree of similarity (or dissimilarity) between the individual objects that are being clustered.

There are two types of clustering methods, *hierarchical* and *partitional.* Hierarchical methods construct successive clusters using previously constructed clusters, while partitional methods determine all clusters at once. Hierarchical methods can be *agglomerative* or *divisive.* Agglomerative methods use the 'bottom-up' approach; they begin with each object as a separate cluster and merge them into successively larger clusters. Divisive methods, on the other hand, use the 'top-down' approach; they begin with the whole set of objects in one cluster and proceed to divide it into successively smaller clusters. Conventional clustering methods construct crisp clusters, meaning that each object belongs to exactly one cluster. An alternative, called *fuzzy* clustering (Dunn, 1973), is that each object can belong to more than one cluster, and associated with each object is a set of membership levels. These indicate the strength of the association between that object and a particular cluster (Zadeh, 1965).

Each object in clustering is described using some language and, as in predictive modeling, the attribute-value description is the most commonly used. The (dis)similarity of objects can then be determined from the pairwise dissimilarities of the values of each of their attributes. The pairwise dissimilarity between attribute values is most often the squared distance for numeric attributes. For nominal attributes, it is most often defined as being zero if the values of both objects are equal, and one otherwise; however, in general, we can have a dissimilarity matrix in which every element defines a pairwise dissimilarity between the possible values of the specific attribute. The dissimilarities along single attributes are then combined into a single overall measure of object dissimilarity. Most often, this is done using a weighted average. In addition, each cluster of objects can also be represented by a prototypical object or a *prototype,* which can be defined as the point in the attribute space with the lowest average dissimilarity (distance) to all objects in the cluster. In this way, we can also define the dissimilarity between two clusters of objects as the dissimilarity between their prototypes.

When comparing clustering to predictive modeling, there are several important differences. Most important, in clustering, the objects usually have no target property to be predicted, and consequently, the encoded examples have only a description part and no target part. The second important difference is that the final result of clustering is most often just a set of object clusters without symbolic descriptions. Each cluster can be, however, considered a separate concept, and symbolic descriptions can be added to already constructed clusters; this approach is called *conceptual clustering* (Michalski, 1980). Despite these differences, there are also several similarities, especially between clustering and predictive modeling methods that partition the space of examples into subsets such as decision trees. Based on these similarities, an approach called *predictive clustering* has been developed, which we describe in the next section.

2.3 Predictive clustering

Predictive modeling and clustering are, because of their many differences, usually regarded as completely different machine learning techniques. However, as we have already mentioned, there are also several similarities between the two techniques. In particular, the predictive modeling methods that partition the examples into subsets, e.g., decision trees and decision rules, can also be viewed as clustering methods (Langley, 1996). Namely, a decision tree can be regarded as a hierarchy of clusters, where each node is a cluster; such a tree is called a *clustering tree* (Blockeel, 1998). Likewise, a decision rule can represent a cluster of examples which it covers. The benefit of using these methods for clustering is that, in addition to the clusters themselves, we also get symbolic descriptions of the constructed clusters. Every cluster in a tree has a symbolic description in the form of a conjunction of conditions on the path from the root of the tree to the given node, and every cluster represented by a rule is described by the rule's condition. There is, however, a difference between 'tree' clusters and 'rule' clusters. 'Tree' clusters are ordered in a hierarchy and do not overlap, while 'rule' clusters in general are not ordered in any way (they are flat) and can overlap (one example can belong to more than one cluster). We can say that clustering trees are a hierarchical clustering method, and *clustering rules* are a partitional (and possibly fuzzy) clustering method.

Predictive clustering (Blockeel, 1998; Blockeel et al., 1998) combines aspects from both predictive modeling and clustering. When the tree based representation is used, we are talking about *predictive clustering trees* (Blockeel et al., 1998). Decision trees partition the set of examples into subsets in which examples have similar values of the target variable, while clustering produces subsets in which examples have similar

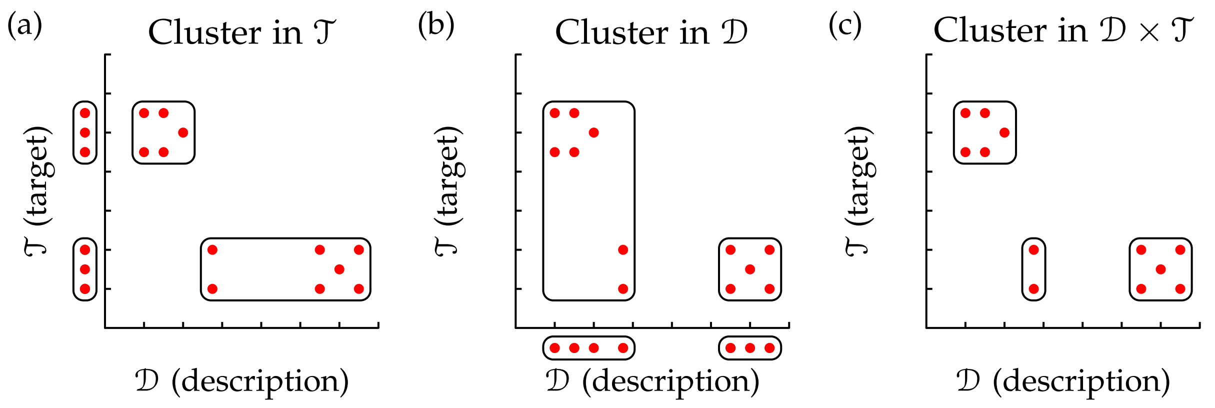

(a)
Cluster in $\mathcal{T}$
$\mathcal{T}$ (target)
$\mathcal{D}$ (description)
(b)
Cluster in $\mathcal{D}$
$\mathcal{T}$ (target)
$\mathcal{D}$ (description)
(c)
Cluster in $\mathcal{D} \times \mathcal{T}$
$\mathcal{T}$ (target)
$\mathcal{D}$ (description)

Figure 2.1: Illustration of predictive modeling (a), clustering (b), and predictive clustering (c). Figure taken from (Blockeel, 1998).

values of the descriptive variables. The task of predictive clustering is to find clusters of examples which have similar values of both the target and the descriptive variables. An illustration of all three cases is given in Figure 2.1 (taken from (Blockeel, 1998)). Let us assume that each example has a description part ($D \in \mathcal{D}$) and a target part ($T \in \mathcal{T}$). $\mathcal{D}$ and $\mathcal{T}$ are descriptive and target attribute spaces, and are in this figure represented as one-dimensional axes, but can both be of higher dimensionality. A (predictive) decision tree method constructs a tree with nodes that comprise examples with similar values of the target value only, i.e., it finds clusters that are homogeneous in the target attribute space $\mathcal{T}$ as shown in Figure 2.1.a. The reason is that the quality criterion that is used to construct the tree (e.g., information gain (Quinlan, 1993)) is based on the target attributes only.[2] In clustering, on the other hand, there are no target attributes defined and the clusters that are generated are homogeneous in the descriptive attribute space $\mathcal{D}$, as shown in Figure 2.1.b. Predictive clustering can now be defined as a method which searches for clusters that are homogeneous in both the descriptive attribute space $\mathcal{D}$ and the target attribute space $\mathcal{T}$ (Figure 2.1.c).

An important part of predictive clustering is its predictive aspect. Namely, each cluster is associated with a predictive model, which gives a prediction of the target variables T in terms of the descriptive variables D for all examples that belong to that cluster. In the most common case, the predictive model associated to a cluster is the projection of the prototype of the cluster on the target attribute space T, i.e., the target part of the cluster's prototype. Typically, the prototype of a cluster is the vector of prototypes of the individual attributes.[3] The prototype of an individual numeric

[2]Because the nodes of a decision tree have conjunctive descriptions in $\mathcal{D}$, they comprise examples that also have somewhat similar values of the descriptive attributes. However, this similarity is not explicitly optimized.

[3]In other words, this means that we are using the Manhattan distance for the dissimilarity measure.

attribute is simply the average of its values in the examples belonging to the cluster, whereas the prototype of a nominal attribute is the probability distribution across its possible values.

Methods for predictive clustering enable us to construct models for predicting multiple target variables,[4] which are normally simpler and more comprehensible than the corresponding set of models, each predicting a single variable. Because similarity estimation is done while taking into account all the variables, predictive clustering is especially suitable for analysis of noisy domains and domains with missing values for the target variables (Blockeel, 1998). So far,[5] this approach has been limited to tree learning methods. The goal of this thesis is to extend predictive clustering with methods for learning rules, and rule learning is briefly presented in the next section.

2.4 Rule learning

Predictive models can be written in more or less understandable forms. Among them, sets of 'if-then' rules are one of the most expressive and human readable model representations (Flach and Lavrač, 2003; Mitchell, 1997). When compared to decision trees, rules are generally considered to be more understandable. Each rule, namely, represents an independent piece of knowledge that can be interpreted separately without other rules from the rule set,[6] while the decision rule has to be interpreted as a whole.

A rule set consists of rules of the form *'IF condition THEN prediction'*; in addition, the rule set usually also has a *default rule*, which is used for prediction of examples that do not satisfy the condition of any other rule from the rule set. The examples that satisfy the *condition* are said to be *covered* by this rule. In the case the examples are represented in attribute-value form, the rule's condition is a conjunction of atomic conditions written in terms of the descriptive attributes. The values of the target attributes of examples covered by a rule can be predicted by the *prediction* of this rule.

A rule set usually consists of more than just one rule, and the example whose target attributes values we wish to predict can in principle be covered by more than one rule. In this context, we distinguish *ordered* and *unordered* rule sets. Ordered rule sets, also called *decision lists*, assume that the rules are ordered in a list, and the rule that covers a given example and comes first in this list is used for predicting the target

[4]There exist also other approaches to multiple target learning, such as *multi-task learning* of neural networks (Caruana, 1997), and stochastic *vector decision trees* (Šprogar et al., 2000)

[5]Independently and in parallel to the research presented in this thesis, Curk et al. (2006) presented a method that also uses the predictive clustering approach for learning rules.

[6]This is less true for ordered rule sets, which is also the reason why unordered rule sets are usually preferred.

attribute values of this example. The rules that also cover the example, but come later in the list, have no influence on the prediction for this example. In unordered rule sets, on the other hand, all the rules that cover a given example participate in the prediction for this example. Since each rule can give a different prediction, a combining scheme, such as weighted voting, is needed for merging the predictions of all rules that cover the example.

There has been a lot of interest in rule learning within the machine learning and statistics communities. Most of the rule learning methods originate in the *AQ* series of methods (Michalski, 1969; Michalski et al., 1986), which all employ the *sequential covering algorithm.* The main problem of the covering algorithm, however, is that it was originally designed for two-class (binary) classification problem domains, and its extension towards more general problem domains is nontrivial. In addition, the rule sets produced by the covering algorithm are by nature ordered, unless rules for only one class value are constructed. We will now briefly describe the covering algorithm as it is implemented in the CN2 method, which is one of more commonly used rule learning methods.

Learning rules with CN2

The *CN2* (Clark and Niblett, 1989; Clark and Boswell, 1991) is an algorithm that iteratively constructs classification rules. In each iteration, a single rule is constructed using a heuristic beam search. When a good rule has been found, it is added to the rule set, and all the examples that are covered by this new rule are removed from the learning set. This process, which is usually denoted as the *sequential covering algorithm*, is repeated until no more good rules can be found or the learning set is empty.

The space of possible conditions that CN2 searches consists of conjunctions of descriptive attribute tests. The construction of rules is done in a general-to-specific fashion. It starts with the most general condition which covers all learning examples. At each stage in the search, CN2 keeps a limited set, or a *beam*,[7] of best conditions found so far. In the next stage of search, it only considers specializations of partial conditions in the beam. Specialization is done by adding attribute tests. The prediction of each rule is the most common class value of the examples covered by the rule. Each candidate rule is evaluated with the heuristic evaluation function. The original version of CN2 (Clark and Niblett, 1989) used entropy as the heuristic evaluation function, while the later versions used accuracy as estimated by the Laplace relative frequency estimate (Clark and Boswell, 1991) or the *m-estimate* (Cestnik, 1990;

[7]The beam is called a *star* in the original terminology.

Džeroski et al., 1993).

Additionally, CN2 can test the significance of each rule as it is constructed. The rule is considered to be significant, if it covers such a pattern of examples that it is unlikely to have occurred by chance. To test the significance, the likelihood ratio statistic is used, which measures the difference between the class probability distribution in the set of examples covered by the rule, and the class probability distribution in the entire learning set. If the test suggests that the rule is not significant, the rule is discarded. Empirical evaluation shows that significance testing reduces the number of learned rules, but also slightly reduces predictive accuracy.

The rules constructed in this way are ordered; when used for classification, each rule is tried in order, and the first rule that covers the example is used for classification. If no learned rule covers the example, the final default rule assigns the most common class in the entire learning set to the example. Alternatively, CN2 can also construct unordered rules. In this case, rules are learned iteratively for each possible class value in turn. When a new rule is found, however, only the examples covered by this rule which belong to the specified class are removed from the learning set. Of course, the learning of rules for each class starts with the complete learning set. When using unordered rules for prediction, several rules can cover each example. Associated to each unordered rule is a vector with counts of covered learning examples for each possible class value. The final prediction is obtained by summing the count vectors of all rules that cover a given example, and predicting the class with the largest count.[8]

There exist many improvements of the original CN2 method. Here, we mention just two, which are directly related to the rule learning method presented later in this thesis. Todorovski et al. (2000) propose the use of *weighted relative accuracy (WRAcc)* (Lavrač et al., 1999) as the heuristic evaluation function in the CN2, instead of the accuracy. The weighted relative accuracy heuristic is defined as

$$\mathrm{WRAcc}(\mathit{rule}) = \mathrm{Coverage}(\mathit{rule})\,[\mathrm{Accuracy}(\mathit{rule}) - \mathrm{DefaultAccuracy}]. \tag{2.1}$$

It consists of two parts, thus providing a tradeoff between rule's generality (rule's coverage) and it's *relative accuracy* (the difference between rule's accuracy and the default accuracy of the domain). The empirical evaluation shows that the WRAcc heuristic alone, without the significance testing, greatly reduces the number of learned rules, at the expense of only a small decrease of accuracy of learned rule sets.

The second extension of the CN2 method comes from the field of *subgroup discovery*

[8]This combining scheme is equivalent to using weighted voting for combining rule prototypes (i.e., vectors of relative frequencies), where the rule's weights are the numbers of learning examples covered by each rule.

(Klösgen, 1996). The task of subgroup discovery is to find 'statistically most interesting' subgroups of examples, which are commonly described in terms of rules. In rule sets constructed with the standard covering algorithm, the rules that are learned first, are learned on learning sets with more examples, while the last learned rules are learned using only few learning examples. As a consequence, rules at the beginning have larger coverage and are statistically more significant than the rules learned later on in the process. Because subgroup discovery focuses on statistically interesting (or significant) rules, only the first few rules may be of interest. To increase the number of possibly interesting and significant rules learned, a *weighted covering* algorithm has been introduced (Gamberger and Lavrač, 2002). The weighted covering algorithm, as opposed to the standard covering algorithm, does not remove the examples that are covered by a newly learned rule. Instead, it only reduces the weights of these examples. This way, even the rules learned later in the learning process can be learned on a larger and more representative number of examples, even though the contribution of some examples may be small. The empirical evaluation shows that the use of weighted covering in combination with the above mentioned WRAcc heuristic, results in a reduced number of learned rules that have higher coverage and significance when compared to the original CN2 method (Lavrač et al., 2004).

Learning regression rules

So far, we have only discussed the learning of classification rules. Unfortunately, there are only a few approaches that can learn regression rules. A brief overview of these follows. Weiss and Indurkhya (1993) have developed a system, called *SWAP1R*, which transforms a regression problem into a classification problem. Target values of the learning examples are grouped in into a set of user-defined bins, which act as class values in the subsequent learning phase. In the learning phase a set of classification rules is learned using the covering algorithm. Originally, the prediction of the target value was computed as the average value of all examples covered by this rule. Later, the authors added the possibility of combining this method with the k-nearest neighbors method (Weiss and Indurkhya, 1995). The idea of transforming a regression problem into a classification one was further developed by Torgo and Gama (1996). They developed a system called *RECLA*, which acts as a generic preprocessor and makes it possible to use an arbitrary classification method to solve regression problems.

A rule learning system that learns regression rules directly was proposed by Torgo (1995). Rules learned with the system called R^2 have linear regression models as the prediction part of the rule, though in principle, many different types of regression models could be used. The system iteratively selects regions in the description attributes space that are still not covered by any rule. Next, a regression model is selected from the space of possible models that has small error in the selected region. In the third step, the rule is specialized by adding conditions with the goal of improving the fit of the rule's regression model. The specialization is guided by an evaluation function that includes both the error and the coverage of the rule. This means that the system searches rules that have low error and cover as many examples as possible.

FORS (Karalič and Bratko, 1997) is an inductive logic programming system for learning regression rules. The learning examples are described in terms of first order logic. Like CN2, it uses the covering algorithm. It iteratively constructs rules; each time a new rule is found, all the examples covered by this rule are removed from the learning set. The procedure is repeated while there are enough examples left. At the end, a default rule is added, if the previous rules do not guarantee the coverage of all examples. As in the R^2 system, the prediction model in each rule can either be a constant or a linear regression model.

Another approach for learning regression rules is the *patient rule induction method (PRIM)* (Friedman and Fisher, 1999). The method seeks subspaces (boxes) in the description attribute space where the average value of the target attribute is high (or low), i.e., it looks for maxima (minima) of the target attribute, the task which is also known as *bump hunting*. The construction of a rule starts with a box containing all learning examples (top down). The original box is then reduced along one of the attributes as to get the maximal mean of the target attribute values within the box. The procedure is repeated until only a minimal number of examples remains within the box. At this point, the process reverses and the box is being expanded, if this increases the box mean. The process of reducing and expanding the box depends on parameters which have to be set in accordance with background knowledge. As a result of this process, we have a series of boxes, each containing a different number of examples. The optimal box size is selected by the user with the help of cross-validation. Now, the examples that are within the selected box (rule) are removed from the learning set, and the process is repeated. The system needs human interaction to set the algorithm parameters in accordance with the background knowledge.

The methods presented so far learn rules directly. An alternative is to first learn a decision tree, and then convert it to a set of rules. The approach is applicable to both classification and regression problems. The problem, however, is that rule sets constructed in this way are relatively large as compared to rule sets learned with rule learning methods. In addition, the conditions in the rules contain many attribute tests which describe the path from the root to the leaf of the tree, and many of these may be redundant. Because of this, rule sets constructed via trees may not be so understandable as rule sets constructed directly. A solution to this problem is post-processing of rules which results in smaller and more understandable rule sets (Quinlan, 1995; Holmes et al., 1999; Friedman and Popescu, 2005).

Chapter 3

Estimating quality of rules and rule sets

Before we can start learning rules and combining them into rule sets, which is the topic of the next chapter, we have to establish the quality criteria that each single rule or the rule set as a whole has to satisfy. In practice, these criteria can be conflicting and we have to either select the most important criterion, or find a suitable compromise between several of them. For example, we usually want the rule set to be small and to have a small error rate, but very small rule sets tend to have high error rates. Additionally, the criteria can be different for different tasks and different problem domains, they can even be regarded as a part of the domain knowledge. In this chapter we present the measures that are used as quality criteria in the process of rule and rule set learning (Chapter 4) and rule set evaluation (Chapter 5).

3.1 Single rules

Each single rule within a rule set should represent a *novel* and *general* piece of *knowledge.* For a rule to be regarded as knowledge, it first has to be accurate, or have a small error rate. As we will show later, error rate can be regarded as a special case of *dispersion.* Second, for a rule to be general, it has to generalize over many learning examples or have a large *coverage.* Third, for a rule to represent novel knowledge, it should not cover examples that other rules from the rule set are already covering, i.e., its *distance to existing rules* should be large. In addition, the prototype[1] of the examples covered by a rule should be different from the prototype of the entire set of examples, since such a rule would not provide any new information. So, *dissimilarity of a rule's*

[1]A prototype is a representation of a set of examples with a single example. It is defined later in this chapter.

prototype and the prototype of the entire learning set should be high. Definitions of these measures follow.

3.1.1 Dispersion

Normally, error measures in machine learning are specifically tailored to either classification or regression single target prediction tasks. But for predictive clustering rules we need a measure that can be used in both classification and regression tasks, and also in multiple target tasks. This means that the measure, we call it *dispersion*,[2] should take as input one or more nominal or numeric variables and return an aggregated numeric value, preferably within the $[0, 1]$ interval.

Let E' be the set of N examples that are covered by a specific rule or cluster

$$E' = \bigcup_{i=1}^{N} \mathbf{e}_i = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}.$$

Each example $\mathbf{e}_i$ is represented as a vector of K attribute values x_{ji}

$$\mathbf{e}_i = [x_{1i}, x_{2i}, \ldots, x_{Ki}],$$

where x_{ji} stands for the value of the attribute a_j of the example $\mathbf{e}_i$. In general, and depending on the background knowledge, some attributes may be more important than others. The weights

$$\mathbf{w} = [w_1, w_2, \ldots, w_K]$$

can be used to emphasize certain attributes; by default all weights can be set to 1.

We define our dispersion measure as a weighted average of the dispersions along each attribute. The dispersion of a set of examples E' is then

$$\operatorname{disp}(E'; \mathbf{w}) = \frac{1}{K} \sum_{j=1}^{K} w_j \operatorname{disp}(E', a_j). \tag{3.1}$$

The definition of dispersion along a single attribute depends on its type. For a nominal attribute, it is the average distance of a single example from a set to the prototype of this set. Let the attribute a_j have L possible values with labels l_1 to l_L. The prototype of a set of examples E' of an attribute a_j is a vector of relative frequencies

[2]In statistics, dispersion (also called variability) is a measure of variation of measurements of a random variable. It is a non-negative real number that is zero if all the data are identical, and increases as the data becomes more diverse. Examples of dispersion measures are standard deviation for numeric and entropy for nominal variables.

f_k of possible values within the set

$$\mathbf{p}_{E'} = \mathbf{p}(E'; a_j) = [f_1, f_2, \ldots, f_L]; \quad f_k = \frac{n_k}{N}; \quad \sum_{k=1}^{L} n_k = N, \tag{3.2}$$

where n_k stands for the number of examples in the set E' whose value of attribute a_j equals l_k. Accordingly, (the prototype of) a single example $\mathbf{e}_i$ with the value of attribute a_j equal to l_k is

$$\mathbf{p}_{\mathbf{e}_i} = \mathbf{p}(\{\mathbf{e}_i\}; a_j) = [f'_1, f'_2, \ldots, f'_L]; \quad f'_k = \begin{cases} 1, & \text{if } x_{ji} = l_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

The distance between the two prototypes can be measured using any of the distance measures defined on vectors; we have decided to use the *Manhattan distance.* Now the distance between an example $\mathbf{e}_i$ with the value of attribute a_j equal to l_k (i.e., the prototype $\mathbf{p}_{\mathbf{e}_i}$) and prototype of the entire set E' is

$$d(\mathbf{p}_{\mathbf{e}_i}, \mathbf{p}_{E'}) = |1 - f_k| + \sum_{\substack{m=1 \\ m \neq k}}^{L} |f_m| = 2(1 - f_k); \tag{3.4}$$

where we have taken into account that $f_k < 1$ and $\sum f_m = 1$. Finally, the dispersion of the set of examples E' along the nominal attribute a_j is the normalized average distance

$$\mathrm{disp}(E', a_j) = \frac{1}{2N} \frac{L}{L-1} \sum_{i=1}^{N} d(\mathbf{p}_{\mathbf{e}_i}, \mathbf{p}_{E'}). \tag{3.5}$$

The normalization factor normalizes the value of dispersion to the $[0, 1]$ interval which is necessary, if we want the dispersions between different attributes to be comparable.

Dispersion along a numeric attribute could be defined in a similar way, using the mean of the attribute's values as the prototype and the absolute difference as the distance between the example and the prototype. However, we have decided to use variance as the dispersion measure for numeric attributes instead, because variance is a standard measure in statistics and is most commonly used in this context. The variance of an attribute can be estimated as

$$s_N^2(E', a_j) = \frac{1}{N} \sum_{i=1}^{N} (x_{ji} - \overline{x}_j)^2, \tag{3.6}$$

where x_{ji} is the value of attribute a_j of example $\mathbf{e}_i$ and $\overline{x}_j$ is the mean of values of attribute a_j in set E'. The variance values are not limited to the $[0, 1]$ interval, so we have to normalize them. The dispersion of the set of examples along one numeric attribute is then

$$\mathrm{disp}(E', a_j) = \frac{1}{\nu_j^2} s_N^2(E', a_j). \tag{3.7}$$

One way of setting the normalization factor ν_j would be the range of values of the attribute a_j in the set of examples E'

$$\nu_j = a_j^{range} = a_j^{max} - a_j^{min}. \tag{3.8}$$

However, the problem with range normalization is that it is very sensitive to noise and outliers. A better solution is to link the normalization factor to the standard deviation of the values of the attribute. If we select

$$\nu_j = 4\,\sigma_{a_j}, \tag{3.9}$$

and if the values of the attribute are normally distributed, we can be sure that most of the values will be within the $[0, 1]$ interval. This is due to the fact that 95% of the values of a normally distributed variable lie within the $[\overline{x}_j - 2\sigma_{a_j}, \overline{x}_j + 2\sigma_{a_j}]$ interval.

3.1.2 Coverage

Each rule should represent a general piece of knowledge. More generalization means that the rule covers more examples and in the end, it also means that the final rule set will have fewer rules and will be more comprehensible. Unfortunately, more generalization most often also means larger error in the model, and a compromise between the two must be found. The definition of relative coverage is straightforward. Let the complete set of examples E have N examples, and let E' be the set of examples that satisfy the condition of rule r, i.e., the set of examples covered by rule r; let $M = |E'|$. The relative coverage of rule r is then

$$\mathrm{cov}(r; E) = \frac{|E'|}{|E|} = \frac{M}{N}. \tag{3.10}$$

The above formula assumes that all examples are equally important, i.e., they all have equal weight. Sometimes, however, it is useful to introduce example weights that are not uniform. Each example $\mathbf{e}_i$ then has an associated weight w_i. The relative coverage of rule r in this case is simply the sum of weights of the examples covered by r divided by the sum of weights of all examples

$$\mathrm{cov}(r; E, \mathbf{w}) = \frac{\sum\limits_{\mathbf{e}_i \in E'} w_i}{\sum\limits_{\mathbf{e}_i \in E} w_i}. \tag{3.11}$$

In case all example weights are equal to one, the above formula reduces to Equation 3.10.

3.1.3 Distance to existing rules

Learning rules is an iterative procedure, we learn one rule after another until a stopping criterion is met. When learning a new rule, we should take into account already learned rules (except when learning the first rule). Namely, we want our rules to be novel, i.e., we do not want to have many rules that cover the same examples, but we want our rules to each cover a different part of the example space. In rule learning, this problem is usually solved by modifying the learning set, as in the case of the covering algorithm. An alternative approach is to define a distance from a given rule to already learned rules in terms of covered examples.

Let E be the learning set with N examples, $R = \{r_k\}_1^{\varrho}$ be the set of existing rules with ϱ rules, and r be the rule we are evaluating. The distance between a rule r and a set of rules R can be defined as the average distance between the rule r, and each rule from the set R

$$\mathrm{dist}(r, R) = \frac{1}{\varrho} \sum_{k=1}^{\varrho} d(r, r_k), \tag{3.12}$$

where the distance between two rules is defined as

$$d(r_j, r_k) = \frac{1}{N} \sum_{i=1}^{N} d_1(r_j, r_k; \mathbf{e}_i), \tag{3.13}$$

and the distance between two rules on example $\mathbf{e}_i$ as

$$d_1(r_j, r_k; \mathbf{e}_i) = \begin{cases} 0, & \text{if both } r_j \text{ and } r_k \text{ cover example } \mathbf{e}_i, \\ 0, & \text{if both } r_j \text{ and } r_k \text{ do not cover example } \mathbf{e}_i, \\ 1, & \text{otherwise.} \end{cases} \tag{3.14}$$

An alternative to the above defined distance measure would be, for example, set similarity measure such as the *Jacquard coefficient*.

3.1.4 Prototype dissimilarity

As already mentioned in Section 3.1, we should learn rules that cover examples with a prototype that is different from the *default prototype*, i.e., the prototype of the whole learning set of examples. Namely, a rule with a prototype equal to the default prototype would give the same predictions as the default rule, which is usually always in the rule set. Predictions of such a rule would not be useful since they would not provide any new information, i.e., their *information score*[3] would be zero (Kononenko and Bratko, 1991). Measuring prototype dissimilarity is in principle similar to the significance testing (or measuring of significance) of rules as implemented in the CN2 algorithm (Clark and Niblett, 1989; Clark and Boswell, 1991). In CN2, the likelihood ratio statistic is used to measure the difference between the class probability distribution in the set of examples covered by the rule, and the class probability distribution of the entire learning set. If the value of the statistic suggests that the class distribution of the rule is not significantly different, the rule is discarded. Originally, the reasoning for significance testing was to avoid adding rules that are due to chance. The prototype dissimilarity measure presented here can be regarded as a generalization of this approach since it can take into account more than one attribute.

We define our prototype dissimilarity measure as a weighted average of the prototype dissimilarities along each attribute

$$\mathrm{diss}(r; E, \mathbf{w}) = \frac{1}{K} \sum_{j=1}^{K} w_j \, \mathrm{diss}(r; E, a_j), \tag{3.15}$$

where r is the rule we are evaluating, E is the entire learning set, and $\mathbf{w}$ is the attribute's weight vector. The attributes, namely, can be of different importance, and assuming some domain knowledge, weights enable us to take this into account.

Prototype dissimilarity along a single attribute is different for nominal and for numeric attributes

$$\mathrm{diss}(r; E, a_j) = \begin{cases} \dfrac{1}{L} d(\mathbf{p}_r, \mathbf{p}_E), & \text{if } a_j \text{ is a nominal attribute,} \\ \dfrac{1}{\sigma_{a_j}} d(p_r, p_E), & \text{if } a_j \text{ is a numeric attribute.} \end{cases} \tag{3.16}$$

Normalization factors are used for limiting the values to the $[0, 1]$ interval, so that the contributions of different attributes are comparable. The d measure can, in principle, be any dissimilarity measure; for simplicity, we have again selected the *Manhattan*

[3]The information score is only defined for single target classification tasks.

distance. Prototypes of examples along a single nominal attribute are vectors with L components, where L is the number of possible values of this attribute; they have already been discused in Section 3.1.1, Equation 3.2. The distance between the prototypes is then

$$d(\mathbf{p}_r, \mathbf{p}_E) = \sum_{k=1}^{L} |p_{rk} - p_{Ek}|. \tag{3.17}$$

Prototypes of examples along a single numeric attribute are themselves numeric values. We can define them as the average value of the attribute a_j

$$p(E'; a_j) = \frac{1}{N} \sum_{i=1}^{N} x_{ji}, \tag{3.18}$$

where E' is the set of N examples whose prototype we are seeking and x_{ji} are the values of attribute a_j in example $\mathbf{e}_i$. The distance between two prototypes of numeric attributes is simplified Equation 3.17

$$d(p_r, p_E) = |p_r - p_E|. \tag{3.19}$$

3.2 Rule sets

There are many measures for rule set evaluation and model evaluation in general that are used in machine learning. In classification tasks, the most commonly used measure is *classification accuracy,* or its complement, *classification error.* Some other standard measures, such as *precision, recall,* or *area under ROC curve* are specifically tailored for binary classification problems and therefore inappropriate for multiple class, let alone multiple target problems. In regression tasks, the common evaluation measures are *relative root mean squared error (RRMSE)* and *correlation coefficient*, but *root mean squared error (RMSE), mean squared error (MSE),* and *mean absolute error (MAE)* are also widely used. The measures presented here are by no means all possible measures that can be used for this purpose, they are a selection of commonly used measures that have been extended towards multiple target tasks. A more extensive preview of quality measures can be found in general textbooks on machine learning or statistics, e.g., (Kononenko, 2005; Hastie et al., 2001). Depending on the set of examples that we use for estimating, for instance, classification error, we are talking about *classification error on learning set* or *classification error on testing set.* Additionally, we can use a cross-validation procedure to get the *cross-validated classification error.*

3.2.1 Classification error

Classification error is a standard quality measure for single target classification tasks. Let E be the set of N examples which we use to estimate the classification error of rule set R. If N_{err} is the number of examples with incorrectly predicted value of target attribute a_j, the classification error for attribute a_j is

$$\mathrm{err}(R; E, a_j) = \frac{N_{\mathrm{err}}}{N}. \tag{3.20}$$

Although this *relative frequency* estimate of the classification error is unbiased, its use as a heuristic function in the search through a large space of possible hypotheses (in our case rules or rule sets) can lead to overfitting. Namely, because of a large number of required error estimations, random hypotheses can be found that have overly optimistic error estimate. The problem can be alleviated by using the *m-estimate* (Cestnik, 1990) or the *extreme value correction (EVC)* (Možina et al., 2006) of probability.

A faithfull extension of the above classification error measure (Equation 3.20) towards multiple target problems is not straightforward. Here we present just a simple solution, weighted average classification error over all target attributes

$$\overline{\mathrm{err}}(R; E, \mathbf{w}) = \frac{1}{T} \sum_{j=1}^{T} w_j \, \mathrm{err}(R; E, a_j), \tag{3.21}$$

where T is the number of target attributes. We have added attribute weights w_j here for generality and they can be used to put more emphasis on certain target attributes. The most serious objection to this type of averaging is that we are basically adding apples and oranges, meaning that the classification errors of different target attributes are not comparable quantities (see e.g., (Demšar, 2006) for a more in-depth discussion of the topic). Still, we believe that the use of this measure is justified, as long as we are aware of its deficiencies.

3.2.2 Relative root mean squared error

In contrast to classification, where classification error is by far the most important quality measure, in regression, we have a set of commonly used measures. The values of most of them are not even remotely comparable across different problem domains; they change with the scale of variable values. Among the few exceptions are the *relative root mean squared error (RRMSE)*[4] and the *correlation coefficient*. The *root mean*

[4]The relative root mean squared error (RRMSE) can also be called *root relative squared error (RRSE)*, however, we decided to use the former name since it more clearly expresses the fact that this is a

squared error (RMSE) of the attribute a_j is defined as

$$\mathrm{RMSE}(R; E, a_j) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{x}_{ji} - x_{ji})^2}, \tag{3.22}$$

where $\hat{x}_{ji}$ is the value predicted by rule set R, x_{ji} is the true value of the attribute a_j of example $\mathbf{e}_i$, and N is the number of examples. The value of RMSE depends on the actual values of domain variables and their scales. In order to make it independent of these, we normalize it with the RMSE of the default model, i.e., the model that always predicts the average value of the attribute. This gives rise to the relative root mean squared error

$$\mathrm{RRMSE}(R; E, a_j) = \sqrt{\frac{\sum_{i=1}^{N} (\hat{x}_{ji} - x_{ji})^2}{\sum_{i=1}^{N} (\overline{x}_j - x_{ji})^2}}, \tag{3.23}$$

where $\overline{x}_j$ is the mean value of the attribute a_j on the set of examples E.

Everything said about extending classification error towards multiple target problems also holds for RRMSE; the weighted average over all target attributes is defined as

$$\overline{\mathrm{RRMSE}}(R; E, \mathbf{w}) = \frac{1}{T} \sum_{j=1}^{T} w_j \, \mathrm{RRMSE}(R; E, a_j), \tag{3.24}$$

where T is the number of attributes and $\mathbf{w}$ is their weight vector.

3.2.3 Correlation coefficient

The *correlation coefficient,* also referred to as *Pearson's correlation coefficient,* is a measure of linear relation between two variables. If one variable refers to the true values of the target attribute a_j (denoted as x_{ji}) and another refers to the values of the attribute as predicted by the model (denoted as $\hat{x}_{ji}$), the correlation coefficient measures the correlation between the true and the predicted values. The coefficient is equal to the covariance between the two variables divided by the product of their variances

$$r(R; E, a_j) = \frac{s^2_{N-1}(E, a_j, \hat{a}_j)}{\sqrt{s^2_{N-1}(E, a_j) \; s^2_{N-1}(E, \hat{a}_j)}}. \tag{3.25}$$

'relative RMSE'.

The covariance and variances are estimated as

$$s^2_{N-1}(E, a_j, \hat{a}_j) = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ji} - \overline{x}_j)(\hat{x}_{ji} - \overline{\hat{x}}_j), \tag{3.26}$$

$$s^2_{N-1}(E, a_j) = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ji} - \overline{x}_j)^2, \tag{3.27}$$

$$s^2_{N-1}(E, \hat{a}_j) = \frac{1}{N-1} \sum_{i=1}^{N} (\hat{x}_{ji} - \overline{\hat{x}}_j)^2, \tag{3.28}$$

where E is the set of N examples. The values of the correlation coefficient are limited to the $[-1, 1]$ interval, with a value of 1 meaning a perfect linear relation between the values of the two variables, and a value of 0 meaning no correlation at all.

On multiple target problems we can, as for classification error and RRMSE, compute the weighted average correlation coefficient as

$$\overline{r}(R; E, \mathbf{w}) = \frac{1}{T} \sum_{j=1}^{T} w_j \, r(R; E, a_j). \tag{3.29}$$

Of course, we should be aware of the shortcomings of such an estimate.

3.2.4 Complexity

The complexity of a model, in our case, the complexity of a rule set, can be directly linked to its comprehensibility. It is generally true that the smaller the complexity of a model, the better are the chances that humans will be able to understand it and interpret its meaning. The most straightforward measure of the complexity of the rule set R is its size, i.e., the number of rules it comprises

$$\mathrm{cplx}_1(R) = |R|. \tag{3.30}$$

However, not all rules are the same, and some have many tests, while others can have just one. Obviously, a rule with more tests has a larger complexity than a rule with a single test. An alternative, or additional, rule set complexity measure can be the number of all tests in the rule set

$$\mathrm{cplx}_2(R) = \sum_{r_i \in R} n_{r_i}, \tag{3.31}$$

where n_{r_i} is the number of tests in rule r_i.

3.3 Summary

In this chapter we presented measures for evaluation of single rules and complete rule sets. Rule learning (the topic of the next chapter) is usually a sequential process during which the evaluation of single rules is of decisive importance. Single rule quality measures are used in the learning algorithm as a search heuristic and directly influence the quality of learned rules. Measures for rule set quality are also used in the rule learning algorithm, for example when deciding whether to include the newly learned rule in the rule set or not. In addition, rule set measures are also used in the final evaluation of the learned rule sets as discussed in Chapter 5.

Chapter 4

Learning predictive clustering rules

An algorithm for learning predictive clustering rules is presented in this chapter. Predictive clustering rules are in a way a generalization of ordinary classification and regression rules, so this chapter can be seen as a presentation of a general rule learning algorithm. We start with a definition of the learning task and continue with the top level of the algorithm. Specific aspects of the algorithm, such as learning single rules, combining these rules into a rule set, modification of the learning set between subsequent iterations of the algorithm, and optimization and interpretation of the final rule set, are discussed in separate sections. At the end, we summarize the algorithm's parameters and options for selecting specific parts of the algorithm.

4.1 Task definition

Let us define the task of learning predictive clustering rules more formaly. We are given

- a description attribute space $\mathcal{D}$,
- a target attribute space $\mathcal{T}$,
- a set of N examples $E = \{\mathbf{e}_i \mid \mathbf{e}_i = [\mathbf{x}_i, \mathbf{y}_i] \in \mathcal{D} \times \mathcal{T}\}$,
- a declarative language bias B over $\mathcal{D}$,
- a distance measure $d : (\mathcal{D} \times \mathcal{T})^2 \to \mathbb{R}$ that computes the distance between two examples, and
- a prototype function $p : 2^{(\mathcal{D} \times \mathcal{T})} \to \mathcal{D} \times \mathcal{T}$ that computes the prototype of a set of examples.

The attribute spaces $\mathcal{D}$ and $\mathcal{T}$ are Cartesian products of the respective attributes' ranges

$$\mathcal{D} = \bigtimes_{j=1}^{n_d} D_j, \quad \mathcal{T} = \bigtimes_{j=1}^{n_t} T_j,$$

where n_d is the number of *descriptive,* and n_t the number of *target attributes.* Attribute ranges are real numbers for numeric attributes, and sets of labels for nominal attributes

$$D_j, T_j = \begin{cases} \mathbb{R}, & \text{if } a_j \text{ is a numeric attribute} \\ \{l_1, l_2, \ldots, l_{L_j}\}, & \text{if } a_j \text{ is a nominal attribute with } L_j \text{ possible values.} \end{cases}$$

A declarative language bias B defines the language in which hypotheses are described and the space of hypotheses that the learning algorithm can consider. Basically it defines a *predictive clustering rule (PCR)* as a rule of the form

IF *<cluster description>* THEN *<cluster prototype>*,

where the cluster description (or rule condition) is a conjunction of tests on descriptive attributes or *"true"*, if the cluster contains all examples (i.e., the cluster description is empty)

$$\textit{<cluster description>} = \bigwedge_k t_k \mid \textit{``true''},$$

and each test can be one of the following ($a_j \in D_j$)

$$t_k = \begin{cases} a_j \leq c_k, & \text{if } a_j \text{ numeric attribute,} \\ a_j > c_k, & \text{if } a_j \text{ numeric attribute,} \\ a_j \in L_k, & \text{if } a_j \text{ nominal attribute with } L_j \text{ possible values.} \end{cases}$$

Here c_k is some numeric constant and L_k is a proper subset of all possible values of attribute a_j

$$L_k \subset \{l_1, l_2, \ldots, l_{L_j}\}.$$

Such a definition of L_k means that we are not limited to simple tests of the form $a_j = l_k$ for nominal attributes, but something like $a_j \in \{l_1, l_3, l_4\}$ is also allowed.

The cluster prototype in a predictive clustering rule is the target part of the prototype of examples that satisfy its description. It has the form

$$\textit{<cluster prototype>} = \mathbf{p}_T = [p_{a_x}, p_{a_y}, \ldots, p_{a_z}]; \quad a_x, a_y, \ldots, a_z \in T_x, T_y, \ldots, T_z,$$

where the attributes a_x, a_y, to a_z are the target attributes. A prototype along a single nominal attribute is defined by Equation 3.2, while the prototype of a numeric attribute is the average of its values[1] (See Section 3.1.1 for more details.)

Now the task is to find a set of predictive clustering rules R where each rule represents a cluster of examples. For these clusters we request that

- distances between examples within each cluster are low, and
- distances between examples from different clusters are high.

In other words, we want the clusters to comprise similar examples and, at the same time, different clusters to comprise dissimilar examples. The algorithm for finding such clusters is discussed in the next section.

4.2 Top level algorithm

Some existing approaches to rule learning were discussed in Section 2.4 and we saw that most of them are based on the *covering algorithm* (Michalski, 1969; Michalski et al., 1986). Here we present a more general approach to rule learning, which subsumes the covering algorithm as a special case. The top level of the algorithm is described here, while its details are discussed in subsequent sections.

The algorithm for learning predictive clustering rules[2] is presented in Table 4.1. We start with an empty rule set R and a set of learning examples E. In each iteration we learn a set of *candidate rules* R_c and, if there is any *good rule* r_i among them, we add this rule to the rule set. Next we *modify the current learning set* E_c and, unless some *stopping criterion* is met, repeat the loop. Before the learning procedure is finished, we have to add the *default rule* and, optionally, *optimize the rule set*.

The above brief algorithm description includes several vague expressions (typeset in *italics*) that need to be discussed in more detail. Learning the set of candidate rules is discussed in Section 4.3, for now let us assume that the procedure 'FindCandidateRules' takes as input the current learning set E_c and returns a set of rules R_c that we then consider as potential members of the rule set. If there is a rule in R_c that satisfies certain quality criteria, we add it to the rule set. This part of the algorithm is discussed in Section 4.4. The next step is the modification of the current learning set, which should reflect the influence of the newly added rule, e.g., one can remove the learning examples covered by the new rule, so that these examples do not influence

[1] A possible improvement to this approach would be a definition of numeric attribute prototypes in a form of (linear) equations, but we do not consider such prototypes in this thesis.

[2] An earlier version of this algorithm was presented in (Ženko et al., 2006).

Table 4.1: Top level of the algorithm for learning predictive clustering rules.

E	... initial learning set
E_c	... learning set in the current iteration
R	... set of rules being learned
R_c	... set of candidate rules
r_i	... rule added in the current iteration

```
procedure LearnRuleSet($E$)
    $R = \emptyset$
    $E_c = E$
    repeat
        $R_c = \text{FindCandidateRules}(E_c)$                ← Section 4.3
        $r_i = \text{BestRule}(R_c, R)$                       ← Section 4.4
        if ($r_i \neq \emptyset$) then
            $R = R \cup \{r_i\}$
        $E_c = \text{ModifyLearningSet}(E_c, r_i)$            ← Section 4.5
    until StopLearning($E_c, R, r_i$)
    $R = R \cup \text{DefaultRule}(E)$
    $R = \text{OptimizeRuleSet}(R, E)$                        ← Section 4.6
    return $R$
```

the learning of candidate rules in the next iteration. Some strategies for modifying the learning set, as well as stopping criteria within the 'StopLearning' procedure are discussed in Section 4.5. When the rule learning loop of the algorithm is finished, the default rule has to be added to the rule set. The default rule is a rule with an empty cluster description and is used for examples that are not covered by any other rule. Its cluster prototype can be constructed either from the complete learning set E, or from the modified learning set E_c, but the first possibility is more commonly used. The last step, the optimization of the rule set, is a step where final changes to the rule set are made; it can roughly be compared to the post pruning step in decision tree learning. This topic is discussed in Section 4.6.

4.3 Learning single rules

In each iteration of the learning algorithm in Table 4.1 the procedure 'FindCandidateRules' should return one or more candidate rules, or no rules, if no rule can be found. The procedure takes the current learning set E_c as input, but also has access to the initial learning set E and the set of rules found so far R. One can think of many approaches to the implementation of this procedure, here we present three of them.

Random rules. The set of candidate rules can be randomly generated. While one can argue if random rule generation can be called learning at all, we can nevertheless use the learning set to determine the ranges of the attribute values, especially of the numeric attributes. The random rule generation can be implemented in many different ways, one of the simplest being as follows. Let us choose to generate only one candidate rule at a time. First, as already mentioned, we use the learning set E_c to determine the ranges of attribute values. Then we randomly generate the number of tests in the rule's cluster description, the attributes in these tests, the values for these tests (which are within the determined ranges), and, finally, the cluster prototype of the rule. Of course, the total number of candidate rules generated in this way must be high, in order for this approach to deliver a reasonable rule set in the end. An alternative and 'less random' approach is to randomly generate only the cluster descriptions in the rules, while the cluster prototypes are the prototypes of the learning examples covered by the generated cluster descriptions. Intuitively, the latter approach should give better results.

Rules from trees. Another candidate rule generating approach, that we discuss here only briefly, is the generation of rules via decision tree learning. Every decision tree can also be written as a set of rules, and therefore, we can learn a predictive clustering tree (Blockeel et al., 1998) on the current learning set E_c and transcribe it into a set of rules. This is our set of candidate rules. For this approach to work as part of the learning algorithm for predictive clustering rules, we must make sure that the learning set E_c is changing from one iteration of the algorithm to the next iteration, otherwise the same rule set is generated each time. We have mentioned some approaches that employ decision trees for classification and regression rule learning in Section 2.4. There, however, the entire rule set in usually learned from the learning set in one iteration, e.g., (Quinlan, 1995; Holmes et al., 1999), or rule sets from a predefined number of iterations are joined in one large rule set, e.g., (Friedman and Popescu, 2005).

Heuristic search. Heuristic search is by far the most common method for generating rules in rule learning algorithms. Here we present a *general-to-specific beam search algorithm* which is very similar to the one implemented in the CN2 system (Clark and

Table 4.2: General-to-specific beam search algorithm for finding candidate rules.

E_c	... learning set
n_r	... number of candidate rules to be returned
b_w	... width of the search beam
C_{best}	... set of best n_r conditions
c_{last}	... worst condition in C_{best}
T_p	... set of possible tests
$\mathbf{p}_i$	... prototype of examples covered by condition c_i
$h(c)$	... heuristic function { h^+ | h^* } (larger value is better)

```
procedure FindCandidateRules($E_c$)
    $c_{last}$ = "true"
    $C = C_{best} = \{c_{last}\}$
    while $(C \neq \emptyset)$
        $C_{new} = \emptyset$
        foreach $c \in C$
            foreach $(t \in T_p \wedge t \notin c)$
                $c_{new} = c \wedge t$
                if $(h(c_{new}) > h(c_{last}))$ then
                    $C_{new} = C_{new} \cup \{c_{new}\}$
                    $C_{best} = C_{best} \cup \{c_{new}\}$
                    if $(|C_{new}| > b_w)$ then
                        $C_{new} = C_{new} \setminus \arg\min_{c' \in C_{new}} h(c')$
                    if $(|C_{best}| > n_r)$ then
                        $C_{best} = C_{best} \setminus \arg\min_{c' \in C_{best}} h(c')$
                    $c_{last} = \arg\min_{c' \in C_{best}} h(c')$
        $C = C_{new}$
    $R_{best} = \{(c_i, \mathbf{p}_i) \mid c_i \in C_{best}\}$
    return $R_{best}$
```

Niblett, 1989; Clark and Boswell, 1991). The algorithm is presented in Table 4.2. The input to the procedure is the learning set of examples E_c, and we also have to specify the number of candidate rules n_r to be returned by the procedure, and the width of the beam b_w, i.e. the number of partial rules maintained during the search. A set of n_r best rules (or actually conditions) found so far as evaluated by the heuristic function h is denoted as C_{best}. We start with the most general cluster description or condition (*"true"*) that is satisfied by all examples in the learning set E_c. Now we begin specialization of all conditions in the current set of conditions C by conjunctively adding an extra test. Here we consider all possible tests (T_p) that are not already in the condition that we are specializing. In addition, we only consider conditions that cover at least a predefined *minimal number of examples* μ, which is also a parameter to the learning algorithm. Every specialization is evaluated using the heuristic function h. If any specialization is better than the worst condition in the set C_{best}, we add it to this set and to set C_{new}. We remove the worst conditions, if the sizes of these sets increase over their predefined maximum sizes. When all specializations of the current set of conditions C are examined, the set C becomes set C_{new}, and the search is continued until no better specializations can be found. At the end, the n_r best conditions from the set C_{best} are coupled with the prototypes of examples that they cover and returned as a resulting set of candidate rules.

The crucial part of the algorithm described above is the search heuristic function h. The heuristic function is used for the evaluation of rules under construction and basically leads the search procedure towards rules of the desired quality. Therefore, the heuristic function should reflect the qualities we expect from each individual rule in the rule set. In Section 3.1 we have discussed a number of rule evaluation measures, let us just briefly recall them. First, we want the rules to be accurate and we have proposed a measure called *dispersion* to measure their accuracy. Next, we want the rules to be general, which we can measure with *coverage*. For the rule to be novel, it should not cover examples that other rules from the rule set are already covering, and we have proposed a *distance to existing rules* to measure this. Finally, a rule should be useful, meaning that it should provide some new information. A precondition for this is that the rule covers a part of the example space that is different from the whole learning set; we have proposed a *prototype dissimilarity* measure to evaluate this rule property. Now, we have to combine all four measures into a single heuristic function. There are, of course, many ways to do this, but we will discuss here only two simple possibilities. We can sum up all four measures or multiply them together.

Let c be the condition (cluster description) of rule r that we are evaluating, and E be the set of all learning examples. E_r is the subset of E with examples that satisfy condition c (i.e., are covered by rule r), and R is the set of rules found so far. The additive version of the heuristic function can then be written as

$$h^{+}(c) = [d_{off} - \mathrm{disp}(E_r;\mathbf{w}_a)] + \alpha\,\mathrm{cov}(r;E,\mathbf{w}_e) + \beta\,\mathrm{dist}(r,R) + \gamma\,\mathrm{diss}(r;E,\mathbf{w}_a), \quad (4.1)$$

and the multiplicative version as

$$h^*(c) = [d_{\mathit{off}} - \mathrm{disp}(E_r; \mathbf{w}_a)] \cdot \mathrm{cov}(r; E, \mathbf{w}_e)^{\alpha} \cdot \mathrm{dist}(r, R)^{\beta} \cdot \mathrm{diss}(r; E, \mathbf{w}_a)^{\gamma}. \tag{4.2}$$

In both equations we have introduced several parameters and weights whose meaning is as follows. The parameters α, β, and γ enable us to put more emphasis on one measure or the other. We assume that the dispersion measure is crucial in the evaluation of rules, so the influence of the other three measures is set relative to dispersion. Rules with larger heuristic function values are better.

The *dispersion offset* d_{off} is a parameter introduced analogously with the *weighted relative accuracy (WRAcc)* heuristic (Lavrač et al., 1999; Todorovski et al., 2000) briefly described in Section 2.4 (Equation 2.1). By setting the value of the d_{off} parameter to the *default dispersion* (i.e., the dispersion of the entire learning set E), the first term of Equation 4.1 (and the first factor of Equation 4.2) can be regarded as the relative dispersion loss. In order to emulate the WRAcc heuristic we additionaly have to select the multiplicative version of the heuristic, and set $\alpha = 1$ and $\beta, \gamma = 0$. On the other hand, if we want to emulate the accuracy heuristic from the CN2 method, we set the parameters d_{off}, α, β, and γ to zero.

Another thing we have to discuss are the attribute weights $\mathbf{w}_a$ and example weights $\mathbf{w}_e$. The attribute weight vector $\mathbf{w}_a$ appears as a parameter of dispersion and prototype dissimilarity measures (the other two measures, coverage and rule distance, are attribute independent). Its role is to determine the influence of individual attributes when calculating both measures, and in general, the weights can be set for each attribute differently, based on some domain knowledge. In practice, though, we always divide the attributes into two groups, descriptive and target attributes, and we propose that each group be weighted differently

$$w_{aj} = w_{a_j} = \begin{cases} \tau, & a_j \text{ is a target attribute,} \\ 1 - \tau, & \text{otherwise,} \end{cases} \tag{4.3}$$

where the *target weight parameter* τ should satisfy $0 < \tau \leq 1$. If our sole objective is to learn rules that are as accurate (on the learning set) as possible, then we should set τ to 1. On the other hand, setting τ to less than 1 should lead to rules that cover examples that are more similar with regard to the values of all attributes. Actually, τ is a parameter that can be used to traverse from the predictive modeling setting on one side to the clustering setting on the other.

The coverage part of Equations 4.1 and 4.2 needs also the example weight vector $\mathbf{w}_e$ as a parameter. Namely, it is not necessary for all the examples to have equal weight; by means of example weights we can give preference to selected examples, which should more likely lead to the construction of rules covering these examples. This is the approach also employed by the weighted covering algorithm (presented later in Section 4.5).

4.4 Adding rules to the rule set

While in the previous section we have presented the algorithms for finding candidate rules, it is now time to discuss the criteria by which these candidate rules can be added to the rule set. More technically, we have to provide a description of the 'BestRule' procedure of the top level algorithm in Table 4.1. We limit ourselves to three possible solutions presented in an algorithmic form in Table 4.3.

For a start, let the procedure 'FindCandidateRules' return only one candidate rule

Table 4.3: 'BestRule' procedure for adding rules to the rule set.

```
$R_c$      ... set of $n_r$ candidate rules
$R$        ... set of rules found so far
$M_{add}$  ... rule adding method {"Always" | "If-Better" | "Check-All-If-Better"}
$Q(R')$    ... rule set quality measure $\{\overline{\text{err}}(R'; E, \mathbf{w}_a) \mid \overline{\text{RRMSE}}(R'; E, \mathbf{w}_a)\}$

procedure BestRule($R_c, R$)
    case ($M_{add}$ = "Always")                    # $n_r = 1$, $R_c = \{r\}$
        return $r$
    case ($M_{add}$ = "If-Better")                 # $n_r = 1$, $R_c = \{r\}$
        if ($Q(R \cup \{r\}) < Q(R)$) then
            return $r$
        else
            return $\emptyset$
    case ($M_{add}$ = "Check-All-If-Better")       # $n_r > 1$
        $r_{best} = \emptyset$
        $q_{best} = Q(R)$
        foreach ($r \in R_c$)
            if ($Q(R \cup \{r\}) < q_{best}$) then
                $r_{best} = r$
                $q_{best} = Q(R \cup \{r\})$
        return $r_{best}$
```

at a time ($n_r = 1$). The most obvious option is to always add the candidate rule to the rule set (M_{add}=*"Always"*). This is also the most common approach used in, e.g., the CN2 algorithm (Clark and Niblett, 1989; Clark and Boswell, 1991). On the other hand, one can argue that it does not make any sense to add a rule, unless this rule somehow improves the rule set. In Section 3.2, we have discussed some measures for rule set evaluation that we can use for this purpose. Classification rule sets can be evaluated using the average classification error (Equation 3.21), and regression rule sets can be evaluated using the average relative root mean squared error (Equation 3.24). Here it makes sense to use only the target attributes for the rule set evaluation, i.e., we set the attribute weights ($\mathbf{w}_a$) as

$$w_{aj} = w_{a_j} = \begin{cases} 1, & a_j \text{ is a target attribute,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Now we simply evaluate the rule set with and without the candidate rule and we only add the candidate rule, if it improves the rule set quality (M_{add}=*"If-Better"*). This rather strict criterion can lead to the premature termination of the learning process, if no rule can be found that improves the quality of the rule set. The problem can be somewhat alleviated, if the procedure 'FindCandidateRules' returns more candidate rules ($n_r > 1$). In this case, we can check all candidate rules; if any improves the rule set quality, we add the one that improves it most (M_{add}=*"Check-All-If-Better"*).

Of course, some other strategies for adding rules, and rule set quality measures could also be used. In addition, the candidate rules could also be optimized before being added to the rule set. For example, the I-REP algorithm (Fürnkranz and Widmer, 1994) greedily prunes the candidate rule until further pruning would increase the rule set error, which is estimated on a separate pruning set. The pruned candidate rule is then added to the rule set.

4.5 Modifying the learning set

The next part of the top level algorithm that we need to discuss is the modifying of the current learning set within each iteration, i.e., the procedure 'ModifyLearningSet' of the algorithm in Table 4.1. Modification of the current learning set (E_c) from one iteration to the next should lead to finding different rules within the 'FindCandidateRules' procedure. This part of the algorithm is especially important if the influence of the 'distance to existing rules' part of the heuristic function is small or zero (parameter β in Equations 4.1 and 4.2). Here we present four possibilities for modifying the learning set, they are given in Table 4.4.

Table 4.4: 'ModifyLearningSet' procedure for modifying the current learning set.

E	… initial learning set
E_c	… current learning set
r_i	… newly added rule
w_{ei}	… current weight of example $\mathbf{e}_i$
M_{mod}	… modifying method {*"Std-Covering"* | *"Err-Weight-Covering"* | | *"Sampling"* | *"None"*}
$g(\mathbf{e}_i, r_i)$	… example re-weighting function
ϵ	… covering weight threshold parameter

```
procedure ModifyLearningSet($E_c, r_i$)
    case ($M_{mod}$ = "Std-Covering")
        foreach ($\mathbf{e}_i \in E_c$)
            if ($r_i$ covers $\mathbf{e}_i$) then
                $w_{ei} = 0$
        return $E_c$
    case ($M_{mod}$ = "Err-Weight-Covering")
        foreach ($\mathbf{e}_i \in E_c$)
            if ($r_i$ covers $\mathbf{e}_i$) then
                $w_{ei} = w_{ei} \cdot g(\mathbf{e}_i, r_i)$
            if ($w_{ei} < \epsilon$) then
                $w_{ei} = 0$
        return $E_c$
    case ($M_{mod}$ = "Sampling")
        $E_c$ = RandomSubSample($E$)
        return $E_c$
    case ($M_{mod}$ = "None")
        return $E_c$
```

The most common approach is the covering algorithm (Michalski, 1969; Michalski et al., 1986). We have discussed this algorithm in Section 2.4. The idea is that we put more emphasis on the learning examples that have not yet been adequately covered. This should force the 'FindCandidateRules' procedure to focus on these examples and find rules to describe them. In the original covering algorithm (M_{mod}=*"Std-Covering"*), examples that are already covered by a rule are removed from the current learning set. Rule learning in the next iteration will therefore focus only on examples that have not yet been covered. This approach is used by the CN2 algorithm (Clark and Niblett, 1989; Clark and Boswell, 1991) for the induction of ordered rules.

The weighted covering algorithm (Gamberger and Lavrač, 2002), on the other hand, assigns a weight to each learning example. Instead of removing the covered example completely, weighted covering only decreases its weight. It does this, however, only for examples that have been correctly classified by the newly added rule. The notion of 'correctly classified example' unfortunately only makes sense for single target classification problems. To overcome this limitation, we develop a more general covering scheme, which we call *error weighted covering*, that is applicable to single and multiple target classification and regression problems (M_{mod}=*"Err-Weight-Covering"*). Error weighted covering is similar to 'ordinary' weighted covering, except that the amount by which example's weight is reduced is proportional to the error the newly added rule makes when predicting the example's target attributes' values. The exact weighting scheme is as follows.

Let every learning example $\mathbf{e}_i$ have an assigned weight w_{ei}. At the beginning, the weights of all examples are set to one. Then, whenever a new rule r is added to the rule set, the weight of each covered example $\mathbf{e}_i$ is multiplied by the value of $g(\mathbf{e}_i, r)$, which is defined separately for classification and for regression problems

$$g(\mathbf{e}_i, r) = \begin{cases} g^{cls}(\mathbf{e}_i, r), & \text{nominal target attributes (classification),} \\ g^{reg}(\mathbf{e}_i, r), & \text{numeric target attributes (regression).} \end{cases} \tag{4.5}$$

The multiplier value for classification is further defined as

$$g^{cls}(\mathbf{e}_i, r) = 1 + (\zeta - 1)k^{cls}(\mathbf{e}_i, r), \tag{4.6}$$

where $k^{cls}(\mathbf{e}_i, r)$ is the proportion of correctly classified target attributes of example $\mathbf{e}_i$ by rule r

$$k^{cls}(\mathbf{e}_i, r) = \frac{\text{number of correctly predicted target attributes of } \mathbf{e}_i \text{ by } r}{\text{number of all target attributes}}, \tag{4.7}$$

and ζ is the *covering weight parameter*, which we discuss below. The multiplier value for regression is proportional to ζ and is limited upward to 1, so that example weights can only decrease

$$g^{reg}(\mathbf{e}_i, r) = \begin{cases} \zeta\, k^{reg}(\mathbf{e}_i, r), & k^{reg}(\mathbf{e}_i, r) \leq 1 \\ \zeta, & \text{otherwise.} \end{cases} \tag{4.8}$$

The value of $k^{reg}(\mathbf{e}_i, r)$ is equal to the average normalized absolute error of all target attributes

$$k^{reg}(\mathbf{e}_i, r) = \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{|\hat{x}_{ji} - x_{ji}|}{\sqrt{\mathrm{var}(E, a_j)}}; \tag{4.9}$$

where n_t is the number of target attributes, $\hat{x}_{ji}$ is the value of attribute a_j of example $\mathbf{e}_i$ as predicted by rule r, and x_{ji} is the true value; $\mathrm{var(E, a_j)}$ is the variance of attribute a_j values in the entire learning set E and can be estimated using Equation 3.27.

The *covering weight parameter* ζ enables us, together with the *covering weight threshold parameter* ϵ, to control the pace of removing covered examples from the current learning set. It should take values between 0 and 1. Setting ζ to 0 means that examples, whose target attributes are correctly predicted by rule r, are imediately removed from the current learning set, i.e., their weights are set to zero. Setting ζ to values greater than 0, on the other hand, means that their weights are reduced less rigorously. The parameter ϵ defines the threshold under which the example weights are considered to be too small to be still included in the learning set; when the example weight falls below this value, it is set to zero.

Another approach by which we can modify the current learning set, and consequently achieve diversity of the learned rules, is sampling of the initial learning set (M_{mod}=*"Sampling"*). The most obvious choice for the sampling procedure is *bootstrap sampling* (Efron and Tibshirani, 1994), which is a form of random sampling with replacement. In each iteration, the candidate rules are learned on a different sample of the learning set, which should result in different candidate rules. The approach is somewhat similar to the ensemble method of *bagging* (Breiman, 1996).

The fourth approach to modifying the current learning set is not to modify the learning set during each iteration at all (M_{mod}=*"None"*). Not modifying the learning set only makes sense if the heuristic function within the 'FindCandidateRules' procedure can force the learning of rules that cover examples not yet covered, i.e., the influence of the 'distance to existing rules' part of the heuristic function is non-zero (parameter β in Equations 4.1 and 4.2).

Stopping criteria. The stopping criteria of the top level algorithm, the 'StopLearning' procedure in Table 4.1, are closely linked to the modification of the current learning set, and therefore it makes sense to discuss them at this point. The simplest stopping criterion is the total number of rules in the rule set (χ), but its use is not very convenient since the optimal number of rules differs from one problem domain to another. A better solution is to stop learning when no good rule has been found by the 'BestRule' procedure, i.e., when the rule returned by this procedure is empty. In

Table 4.5: 'StopLearning' procedure with stopping criteria for PCR learning.

E_c	… current learning set
R	… set of rules found so far
r_i	… rule added in this iteration
χ	… maximal number of rules

```
procedure StopLearning($E_c, R, r_i$)
    $s =$ "false"
    if ($|R| < \chi$) then
        $s =$ "true"
    if ($r_i = \emptyset$) then
        $s =$ "true"
    if ($|E_c| = 0$) then        # $|E_c| = \sum w_{ei}$
        $s =$ "true"
    return $s$
```

addition, the learning process should also terminate when the current learning set is empty (or when the weights of all examples are zero). The latter two criteria are also used in that CN2 algorithm (Clark and Niblett, 1989; Clark and Boswell, 1991). For clarity the above mentioned criteria are presented in Table 4.5.

4.6 Interpretation and optimization of the rule set

So far we have discussed the process of learning the rule set, which is actually a process of finding single rules and building them into a coherent rule set. Now that we have a rule set, the question arises of how we interpret this rule set to get predictions of the target attribute's values. There are two major possibilities, where the rules can be regarded as *ordered* or as *unordered.*

Ordered rules. Ordered rules are interpreted in sequence, one by one, and the first rule that covers an example is used to predict the target values. The rest of the rules in the list are discarded. If there is no rule that covers the example, the default rule is used. Ordered rules are also refered to as a *decision list.* The interpretation of rules as ordered is closely linked to the learning algorithm. For the rules to be interpreted as ordered they should be learned using the standard covering approach, i.e., once a rule is added to the rule set, all covered examples are removed from the current learning

Table 4.6: 'OptimizeRuleSet' procedure for (optional) rule set optimization.

```
$E$          ... learning set
$R$          ... rule set
$\mathbf{w}_r$         ... rule weights
$M_{pred}$   ... prediction method {"Ordered" | "Unordered-Cov-W" |
                                   | "Unordered-Uni-W" | "Unordered-Opt-W"}

procedure OptimizeRuleSet($R, E$)
    case ($M_{pred}$ = "Ordered")
        $\mathbf{w}_r = \emptyset$
    case ($M_{pred}$ = "Unordered-Cov-W")
        foreach ($r_i \in R$)
            $w_{ri} = \mathrm{cov}(r; E)$
    case ($M_{pred}$ = "Unordered-Uni-W")
        foreach ($r_i \in R$)
            $w_{ri} = 1$
    case ($M_{pred}$ = "Unordered-Opt-W")
        $\mathbf{w}_r = \mathbf{w}_{opt}(R, E)$
    return $R$
```

set (M_{mod}=*"Std-Covering"* in Table 4.4). The interpretation of rules as ordered has a major disadvantage, however. The rules cannot be interpreted individually, i.e., separately from the rule set, and this seriously decreases their understandability. In the case of ordered rules, no rule set optimization is necessary and the 'OptimizeRuleSet' procedure presented in Table 4.6 does not set any weights to the rules.

Unordered rules. An alternative interpretation is to regard the rules as an unordered set. In this case, more than one rule can cover a given example, which means that the example is at the intersection of the underlying clusters. Each of the rules can be used to obtain a prediction of the target values, and we need a method for combining these predictions into a single one. A method commonly used for this purpose is the weighted voting scheme. Now, the question arises of how we set the weights. A common solution is to set the weight of a rule proportionally to the number of examples it covers (M_{pred}=*"Unordered-Cov-W"*). Here, the underlying assumption is that the rules covering more examples are more reliable and that their votes should have more weight. This is also the approach the CN2 uses (Clark and Niblett, 1989;

Clark and Boswell, 1991). In some cases, however, the small influence of rules with small coverage can be a drawback. For example, this issue arises in *subgroup discovery*, where the task is to find 'statistically most interesting' subgroups or clusters of examples (Lavrač et al., 2004). Here, the interesting clusters are often small and a uniform weighting scheme (M_{pred}=*"Unordered-Uni-W"*), i.e., all weights are equal, is recommended (Lavrač et al., 2004). Another possibility, somewhat computationally expensive, is to derive the weights via optimization (M_{pred}=*"Unordered-Opt-W"*). This is the topic of the next subsection.

Optimizing the rule set. Instead of using predefined rule weights when combining predictions from different rules, we can determine the weights with an optimization procedure. The property that we are trying to optimize is the accuracy of the entire rule set. In addition, through a procedure called *regularization*, we can also achieve that a significant number of weights is set to zero. This, of course, means that these rules can be discarded and the resulting rule set is smaller and more comprehensible. Such an optimization for single target prediction has been introduced by (Friedman and Popescu, 2004, 2005). A brief presentation of the optimization task for multiple target prediction follows.

Each example $\mathbf{e}_i$ consists of a descriptive and a target part $\mathbf{e}_i = [\mathbf{x}_i, \mathbf{y}_i]$. The rule set $R = \{r_k(\mathbf{x})\}_1^K$ consists of K rules which can predict $\mathbf{y}$ from the values of $\mathbf{x}$. The final prediction can be obtained by weighted voting of all the rules

$$\hat{\mathbf{y}} = F(\mathbf{x}; \mathbf{w}_r) = \sum_{k=1}^{K} w_{rk}\, r_k(\mathbf{x}), \tag{4.10}$$

where w_{rk} is the weight of rule r_k. From the learning set we can estimate the rule weights $\mathbf{w}_r$ which minimize the loss function $L(\mathbf{y}, F(\mathbf{x}; \mathbf{w}_r))$

$$\hat{\mathbf{w}}_r = \arg\min_{\mathbf{w}_r} \frac{1}{N} \sum_{i=1}^{N} L\left(\mathbf{y}_i, \sum_{k=1}^{K} w_{rk}\, r_k(\mathbf{x})\right). \tag{4.11}$$

The loss function can be the same as the rule set quality measure used in procedure 'BestRule' (Section 4.4), i.e., the average classification error (Equation 3.21) for classification problems, and the average relative root mean squared error (Equation 3.24) for regression problems. It is well known (Friedman and Popescu, 2004) that, because of the variability of the learning data, Equation 4.11 often provides poor estimates of the true value of $\mathbf{w}_r$, especially in the case where the learning set size is not large compared to the number of rules. A common remedy is to 'regularize' Equation 4.11

by adding a penalty function

$$\mathbf{w}_{opt}(R,E) = \hat{\mathbf{w}}_r(\lambda) = \arg\min_{\mathbf{w}_r} \frac{1}{N}\sum_{i=1}^{N} L\left(\mathbf{y}_i, \sum_{k=1}^{K} w_{rk}\, r_k(\mathbf{x})\right) + \lambda \sum_{k=1}^{K} |w_{rk}|. \tag{4.12}$$

The penalty function (the last term) is a so-called *lasso* penalty. It is independent of the learning data and has a stabilizing influence on the estimated weights. The λ parameter controls the stabilizing effect; larger values provide more stable, but also more deterministic weight estimates. It was shown (Tibshirani, 1996) that larger values of λ produce increased dispersion among the values of $|w_{rk}|$, often with many values being set to zero. In our case, this means that many rules are discarded because their weights are set to zero. This is exactly the effect we want to achieve, i.e., simplify the rule set as much as possible, and, at the same time, preserve or even increase its accuracy.

The authors of (Friedman and Popescu, 2004, 2005) propose a sophisticated optimization procedure for solving the above task for single target domains. In addition, the procedure also finds the optimal value of the λ parameter. Unfortunately, the procedure has not yet been extended to multiple target domains. However, in principle, it should be possible to solve the optimization problem using some general and robust optimization procedure such as *differential evolution* (Price et al., 2005).

4.7 Summary

In this chapter we have presented a general algorithm for learning predictive clustering rules (Table 4.1). The algorithm is composed of building blocks and for most of them we provide several possible implementations. Using these building blocks one can construct a number of different rule learning algorithms. Table 4.7 collects all parameters and options of the algorithm together with their short description. Through these parameters one can select specific parts of the algorithm as well as set their behavior.

Table 4.7: Parameters of the algorithm for learning predictive clustering rules.

Parameter		Possible values, location in the text, and description
h	heuristic type	($h^+ \mid h^*$); Section 4.3, Table 4.2, Equations 4.1 and 4.2; heuristic function used for rule search
M_{add}	rule adding method	(*"Always"* | *"If-Better"* | *"Check-All-If-Better"*); Section 4.4, Table 4.3; method for adding candidate rules to the rule set
M_{mod}	learning set modifying method	(*"Std-Covering"* | *"Err-Weight-Covering"* | *"Sampling"* | *"None"*); Section 4.5, Table 4.4; method for modifying the current learning set
M_{pred}	prediction method	(*"Ordered"* | *"Unordered-Cov-W"* | *"Unordered-Uni-W"* | *"Unordered-Opt-W"*); Section 4.6, Table 4.6; method used when interpreting rules
b_w	beam width	(>1); Section 4.3, Table 4.2; number of the best candidate rules that are kept during the heuristic beam search procedure
μ	minimal number of examples	(>0); Section 4.3; minimal number (or weight) of examples covered by each rule
d_{off}	dispersion offset	(*default dispersion*, $\mathbb{R}$); Section 4.3, Equations 4.1 and 4.2; offset in the dispersion part of the search heuristic
α	coverage heuristic weight	($[0,1]$); Section 4.3, Equations 4.1 and 4.2; weight of the coverage part in the search heuristic
β	'distance to existing rules' heuristic weight	($[0,1]$); Section 4.3, Equations 4.1 and 4.2; weight of the 'distance to existing rules' part in the search heuristic
γ	'prototype dissimilarity' heuristic weight	($[0,1]$); Section 4.3, Equations 4.1 and 4.2; weight of the 'prototype dissimilarity' part in the search heuristic
τ	target attributes weight	($[0,1]$); Section 4.3, Equation 4.3; weight of the target attributes within the search heuristic, non-target attributes are assumed to have weight of $1-\tau$
ζ	covering weight	Section 4.5, ($[0,1]$); Equations 4.6 and 4.8; weight controlling the amount by which weights of covered examples are reduced within the error weighted covering algorithm
ϵ	covering weight threshold	$[0,1]$); Section 4.5, Table 4.4; (value of example weights within the error weighted covering algorithm below which they are set to zero, i.e., removed from the current learning set
χ	maximal number of rules	(>1); Section 4.5, Table 4.5; maximal number of learned rules
λ	regularization parameter	(≥ 0); Section 4.6, Equation 4.12; influence of the regularization in optimization of the rule's weights

Chapter 5

Experimental evaluation

In this chapter, we experimentally evaluate the newly developed methods for induction of predictive clustering rules. First, we define our evaluation methodology and describe problem domains used in the evaluation. Next, we present several groups of experiments, each designed to investigate a different issue. These include a comparison to existing approaches, a comparison of multiple target to single target prediction, and investigation of the influence of some learning algorithm parameters.

5.1 Evaluation methodology and test domains

When evaluating the newly developed methods, we are mainly interested in the predictive error of the learned models. Estimating predictive error of rule sets was discussed in Section 3.2. In this chapter, we confine ourselves to classification error for classification tasks, while for regression tasks, we use the relative root mean squared error (RRMSE) and the correlation coefficient. All error measures are estimated using 10-fold cross-validation. The folds for a specific data set are the same for all experiments, and are, in the case of single target classification, stratified. The error of multiple target prediction tasks is estimated as the average error over all target attributes. Besides the predictive error, we also consider the complexity of models, which we measure as the number of rules.[1] Trees can be converted to rules and their complexity estimated in the same way. When comparing the error rates (and other quality measures) of two algorithms on multiple data sets, one wants to test whether the observed differences are significant. As recommended by Demšar (2006), we do this with the *Wilcoxon signed-rank* test (Wilcoxon, 1945).

All experiments with predictive clustering rules (PCRs) were performed with the

[1]Measuring the complexity of rule sets only as the number of rules in them does not take into account the complexity of each separate rule (i.e., the number of tests within the rule's condition).

Table 5.1: Default settings of the algorithm for learning predictive clustering rules.

Parameter		Default value
b_w	beam width	10
μ	minimal number of examples	2.0
h	heuristic type	h^* *(i.e., product ver.)*
d_{off}	dispersion offset	*default dispersion*
α	coverage heuristic weight	1.0
β	'distance to existing rules' heuristic weight	0.0 *(i.e., not used)*
γ	'prototype dissimilarity' heuristic weight	0.0 *(i.e., not used)*
τ	target attributes weight	1.0 *(i.e., tar. atts. only)*
ζ	covering weight	0.0 *(i.e., remove immediately)*
ϵ	covering weight threshold	0.1
χ	maximal number of rules	1000
λ	regularization parameter	0.0 *(i.e., not used)*

default parameter settings, except where written differently. The default parameter values are presented in Table 5.1. These are set so as to emulate the CN2 and CN2-WRAcc algorithms as closely as possible. Ordered rules were induced with the *learning set modifying method (M_{mod})* set to *"Std-Covering"* and the *prediction method (M_{pred})* set to *"Ordered"*, which corresponds to the CN2 algorithm for learning ordered rules. Unordered rules were induced with the newly proposed error weighted covering algorithm (M_{mod} set to *"Err-Weight-Covering"*) and were interpreted in the same way as unordered CN2 rules (M_{pred} set to *"Unordered-Cov-W"*), i.e., rules are weighted proportionally to their coverage. The *rule adding method (M_{add})* is set to *"Always"* for classification problems, which is also the same as in the CN2 algorithm. The fact that the covering algorithm learns new rules independently of the already learned rules can be problematic in regression tasks (see Section 5.2.4). The preliminary experiments have shown that this problem can be somewhat alleviated by setting the *rule adding method (M_{add})* to *"If-Better"*. This way, a new rule is added to the rule set only if it improves the performance of the whole rule set, which prevents from adding a new rule which is not compatible with the current rule set. Among the two presented heuristic functions (Equations 4.1 and 4.2) we selected the product version (h^*) which together with the *dispersion offset* (d_{off}) set to the default dispersion, and *coverage heuristic weight* (α) set to 1 corresponds to the *WRAcc* heuristic. The *target attributes weight* (τ) is set to 1 which means that the search heuristic takes into account only the target attributes. The *covering weight* (ζ) is set to 0 which means that correctly predicted learning examples are immediately removed from the learning set (like CN2), while others are removed when their weight falls below the *covering weight threshold*

Table 5.2: *Single target classification* data sets and their characteristics: number of examples, percentage of missing values, numbers of nominal and numeric attributes, number of target attributes (here always 1), and number of all attributes.

Data set	# exs	% miss vals	# nom atts	# num atts	# tar atts	# all atts
australian	690	0	8	6	1	15
balance	625	0	0	4	1	5
breast-w	699	0.25	9	0	1	10
bridges-td	102	5.04	4	3	1	8
car	1728	0	6	0	1	7
glass	214	0	0	9	1	10
heart	270	0	6	7	1	14
image	2310	0	0	19	1	20
iris	150	0	0	4	1	5
sonar	208	0	0	60	1	61
soya	683	9.78	35	0	1	36
tic-tac-toe	958	0	9	0	1	10
vote	435	5.63	16	0	1	17
waveform	5000	0	0	21	1	22
wine	178	0	0	13	1	14

(ϵ) which we arbitrarily set to 0.1. The parameters linked to the algorithm features presented in Chapter 4 but not used in the following experimental evaluation are set to zero.

Predictive clustering rules can be applied to (at least)[2] four different predictive tasks: single target classification, single target regression, multiple target classification, and multiple target regression. The descriptions of the problem domains used for evaluation on each of these types of tasks follow below.

5.1.1 Single target classification problems

Single target classification is a standard machine learning task for which many data sets are publicly available. We have selected 15 data sets from the *UCI Machine Learning Repository* (Newman et al., 1998) which are widely used in various comparative studies. A brief overview of the selected data sets and their characteristics is given in Table 5.2.

[2]In principle, predictive clustering rules can also be used for multiple mixed target classification and regression problems, but no evaluation has been done for this type of problems.

Table 5.3: *Single target regression* data sets and their characteristics: number of examples, percentage of missing values, numbers of nominal and numeric attributes, number of target attributes (here always 1), and number of all attributes.

DATA SET	# EXS	% MISS VALS	# NOM ATTS	# NUM ATTS	# TAR ATTS	# ALL ATTS
AUTO-HORSE	203	1.12	8	17	1	26
AUTO-MPG	398	0.22	3	4	1	8
AUTO-PRICE	159	0	0	15	1	16
BREAST-TUMOR	286	0.35	8	1	1	10
CLOUD	108	0	2	4	1	7
CPU	209	0	1	6	1	8
ECHO-MONTHS	130	8.29	3	6	1	10
HOUSING	506	0	1	12	1	14
META	528	4.55	2	19	1	22
PBC	418	16.47	8	10	1	19
QUAKE	2178	0	0	3	1	4
SENSORY	575	0	11	0	1	12
SERVO	167	0	4	0	1	5
STRIKE	625	0	1	5	1	7
VETERAN	137	0	4	3	1	8

5.1.2 Single target regression problems

Single target regression is another standard machine learning task for which many publicly available data sets exist. We have selected 15 data sets from the *UCI Machine Learning Repository* (Newman et al., 1998) and the *StatLib Data Sets Archive* (StatLib). A brief overview of the selected data sets and their characteristics is given in Table 5.3.

5.1.3 Multiple target classification problems

Multiple target classification is a relatively new machine learning task and consequently there are few publicly available data sets. Nevertheless, some data sets from the *UCI Machine Learning Repository* (Newman et al., 1998) can also be regarded as multiple target problems (BRIDGES, MONKS, SOLAR-FLARE, and THYROID). Several problems that are originally multiple target regression problems can be simply discretized (EDM-DIS, SIGMEA-DIS, and WATER-QUALITY-DIS). In addition, we use the data set MEDIANA which is a multiple target classification problem in its original form. An overview of the selected data sets and their characteristics are given in Table 5.4. A brief description of the data sets follows.

Table 5.4: *Multiple target classification* data sets and their characteristics: number of examples, percentage of missing values, numbers of nominal and numeric attributes, number of target attributes, and number of all attributes.

Data set	# exs	% miss vals	# nom atts	# num atts	# tar atts	# all atts
bridges	102	6.05	4	3	5	12
edm-dis	154	0	0	16	2	18
mediana	7953	0	21	58	5	84
monks	432	0	6	0	3	9
sigmea-real-dis	817	0	0	6	2	8
sigmea-sim-dis	10368	0	2	9	2	13
solar-flare-dis	323	0	10	0	3	13
thyroid-0387	9172	4.56	22	7	7	36
water-quality-dis	1060	0	0	16	14	30

BRIDGES

The *Pittsburgh Bridges* data set from the *UCI repository* contains 102 bridges described in terms of 7 attributes (e.g., year of construction, purpose, number of lanes, etc.) and each bridge has 5 properties or target attributes (e.g., construction material, span, type of construction, etc.) which we try to predict.

EDM-DIS

The *Electrical Discharge Machining* is originally a two target regression problem and is described in Section 5.1.4. Both target attributes have been discretized to three nominal values; '*-1*' for negative values, '*1*' for positive values and '*0*' otherwise.

MEDIANA

The data on the Slovene media space were collected by the *Institute for Market and Media Research, Mediana* (www.mediana.si). The data set consists of 7953 questionnaires tracking all important printed, audio, and visual mass media in Slovenia. Originally, each questionnaire contains about 1200 questions about a person's relation to specific media, their activities, interests, and lifestyle, as well as demographic data. The data set has been used within the European project *Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (SolEuNet)*, and some results of the analysis of these data are presented in (Škrjanc et al., 2001). From the complete data set we have selected 79 descriptive attributes from which we try to predict whether a person reads each of the five major Slovenian daily newspapers (*Delo,* *Dnevnik, Večer, Slovenske novice,* and *Ekipa*). These are the target attributes with possible values *'yes'* and *'no'*. The descriptive attributes describe the person's spare time interests (e.g., reading books, going to the cinema, doing sports, etc.), media interests (e.g., science, politics, nature, etc.), their ability to recognize trademarks (of, e.g., cars, beers, washing powders, etc.), whether they own specific items (e.g., apartment, personal computer, mountain bike, etc.), and some other demographic data (e.g., education level, income, age, etc.). Most of the descriptive attributes have either numeric values or nominal values graded on a *Likert scale* (Likert, 1932) with 2, 5, or 9 levels.

MONKS

The *Monks* problems are based on an artificial robots domain (Thrun et al., 1991). Each example is a robot described with 6 nominal attributes *(a1, a2, ..., a6)* each having between 2 and 4 possible values. There are 432 possible robots and three classes, defined as follows:

$$\begin{aligned}
\textit{monk-1} &= (a1 = a2) \lor (a5 = 1) \\
\textit{monk-2} &= \textsc{exactly two of}\ \{a1 = 1, a2 = 1, a3 = 1, a4 = 1, a5 = 1, a6 = 1\} \\
\textit{monk-3} &\simeq ((a5 = 3) \land (a4 = 1)) \lor ((a5 \neq 4) \land (a2 \neq 3)).
\end{aligned}$$

To the last class *(monk-3)*, 5% of noise has been added. A robot either belongs to a class or not. Each of the monk's tasks is therefore is a binary classification task with 3 target attributes.

SIGMEA-REAL-DIS

The *Sigmea Real* is originally a two target regression problem and is described in Section 5.1.4. Both target attributes have been discretized to two nominal values; *'1'* for positive values and *'0'* for zero values.

SIGMEA-SIM-DIS

The *Sigmea Simulated* is originally a two target regression problem and is described in Section 5.1.4. Both target attributes have been discretized to two nominal values; *'1'* for positive values and *'0'* for zero values.

SOLAR-FLARE-DIS

Solar Flares are sudden and intense variations in the brightness of the Sun. In this data from the *UCI repository*, each flare producing region of the Sun is one example. Its status within the previous 24 hours is described in terms of 10 attributes (e.g., area of the region, area of the largest Sun spot, activity, etc.). The task is to predict the flares production (i.e., the number of flares) of each class (*C – common, M – moderate,* or *X – severe* flares) in the following 24 hours. The numbers of flares of each class are the target attributes. Because the *common* flares are much more common than, say, *severe*, the distribution of the classes is highly skewed. The data were collected between February 13 and March 27, 1969.

THYROID-0387

In the *UCI repository* one can find several variants of the *Thyroid Disease* data set. We are using the one with the label *0387* which consists of 9172 patient records from 1984 to early 1987 and originates from the Garvan Institute of Sydney, Australia. For each patient we have given her or his condition and treatment history in the form of 29 attributes. The task is to diagnose 7 different conditions for each patient. Each of these conditions is one target attribute.

WATER-QUALITY-DIS

The *Water Quality* data set is originally a 14 target regression problem and is described in Section 5.1.4. The target attributes have been discretized to two nominal values; *'1'* for positive values and *'0'* for zero values (abundances have been transformed into presences or absences).

5.1.4 Multiple target regression problems

Just like multiple target classification, multiple target regression is a relatively new machine learning task, and there are few publicly available data sets. In fact, we could only find one data set in the *UCI Repository* that can be regarded as a multiple target regression data set (SOLAR-FLARE). All the other data sets used here are not publicly available. A description of each of the data sets follows.

EDM

Electrical Discharge Machining (EDM) is a machining method primarily used for hard materials that are electrically conductive. The workpiece surface is machined by electrical discharges occurring in the gap between two electrodes, the tool and the workpiece. The gap is continuously flushed by the dielectric fluid. The process consists of numerous randomly ignited monodischarges generating crater-textured surface. The

Table 5.5: *Multiple target regression* data sets and their characteristics: number of examples, percentage of missing values, numbers of nominal and numeric attributes, number of target attributes, and number of all attributes.

DATA SET	# EXS	% MISS VALS	# NOM ATTS	# NUM ATTS	# TAR ATTS	# ALL ATTS
EDM	154	0	0	16	2	18
MICROARTHROPODS	1944	5.3×10^{-2}	0	142	3	145
SIGMEA-REAL	817	0	0	6	2	8
SIGMEA-SIM	10368	0	2	9	2	13
SOLAR-FLARE	323	0	10	0	3	13
WATER-QUALITY	1060	0	0	16	14	30

stability and quality of the process depend on many parameters such as the work-piece material, dielectric fluid type, size of the gap between the electrodes, flow of the dielectric fluid, electric voltage between the electrodes, electric current between the electrodes, characteristics of discharges, and others. While these parameters have some predefined values, machining time can often be shortened by their dynamic control. A human operator is normally employed for this purpose. The task is to automatically reproduce the operator's behavior. The data set describes 154 actions taken by the operator where he controlled two variables (target attributes), the flow and the gap. For each variable, three actions were possible: he increased the variable (numerical value 1), decreased the variable (value -1), or took no action (value 0). The reasons for each action are described in terms of 16 numeric attributes which define the type and size of electrical pulses and their history (e.g., mean value and standard deviation of effective pulses within the last 5 seconds). A more detailed description of the data, as well as their analysis can be found in (Karalič and Bratko, 1997).

MICROARTHROPODS

In agricultural soil, a suite of anthropogenic events shape the ecosystem processes and populations including factors like crop and tillage practices. This data set describes agricultural events (e.g., crops planted, packing, tillage, fertilizer and pesticide use, etc.) and soil biological parameters (e.g., abundances of various species of microarthropods) for each soil sample. The task is to induce a model that predicts soil quality from agricultural measures and events. Soil quality is described in terms of *acari* and *collembolan* species, as well as *Shannon biodiversity*. The data have previously been analyzed in (Demšar et al., 2006), where a detailed description can also be found.

SIGMEA-REAL

The large scale release of *Genetically Modified Organisms (GMOs)* raises many agronomic and ecological concerns. The primary concern is to limit the uncontrolled escape of the introduced genes from the genetically modified organisms. A key process to assessing the risks of genetically modified crops, such as oilseed rape, is the study of dispersal capacity of pollen grains. The data were collected within the European project *Sustainable Introduction of GMOs into European Agriculture (SIGMEA)*. A field experiment was conducted, where a pollen donor plot of 10m by 10m was sown with the herbicide resistant, male-fertile (MF) transgenic line of oilseed rape. The donor plot was surrounded by an area of 90m by 90m, sown with two lines of oilseed rape: the herbicide susceptible, near-isogenic MF line, and the non-transgenic male-sterile (MS) line. Seeds were sown on every node of a 29 by 29 grid throughout the field. Later, the seeds of MS and MF plants from the surrounding area were harvested separately on each of the nodes in the grid, sown again, and the seedlings tested on herbicide resistance. The herbicide resistance rate is used as a measure of pollen dispersal. The task is to model the rate of herbicide resistance of the two lines of plants (MF and MS) in dependence of the position, cardinal direction and distance from the center of the donor field, the visual angle between the sampling plot and the donor field, and the shortest distance between the plot and the nearest edge of the donor field. Each plot in the grid represents one example in the data set. The data have previously been analyzed in (Demšar et al., 2005), where a detailed description can also be found.

SIGMEA-SIM

Just like the SIGMEA-REAL data set, this data set is also concerned with gene flow between the genetically modified and conventional oilseed varieties, and it also originates from the SIGMEA project. It was not, however, collected during a field experiment, but was generated using the *GeneSys* model. GeneSys (Colbach et al., 2001a,b) is a complex numeric model that quantifies the effects of cropping systems (crop distribution, crop succession, cultivation techniques oilseed varieties) on gene flow between oilseed varieties and volunteers in time and space. The task is to investigate the effects of the individual field characteristics and cropping systems on pollen and seed dispersal. A set of different individual fields with different cropping systems was generated. Field characteristics were varied over a large range of values, and different cropping systems were selected in order to form contrasted situations of gene flow, ranging from maximum risk to risk free. GeneSys was used to simulate the

Table 5.6: The attributes of the WATER-QUALITY and WATER-QUALITY-DIS data sets.

INDEPENDENT ATTRIBUTES PHYSICAL & CHEMICAL PROPERTIES	TARGET ATTRIBUTES ABUNDANCE OF TAXA
WATER TEMPERATURE	CLADOPHORA SP.
ALKALINITY (PH)	GONGROSIRA INCRUSTANS
ELECTRICAL CONDUCTIVITY	OEDOGONIUM SP.
DISSOLVED O_2	STIGEOCLONIUM TENUE
O_2 SATURATION	MELOSIRA VARIANS
CO_2 CONC.	NITZSCHIA PALEA
TOTAL HARDNESS	AUDOUINELLA(CHANTRANSIA) CHALYBEA
NO_2 CONC.	ERPOBDELLA OCTOCULATA
NO_3 CONC.	GAMMARUS FOSSARUM
NH_4 CONC.	BAETIS RHODANI
PO_4 CONC.	HYDROPSYCHE SP.
CL CONC.	RHYACOPHILA SP.
SIO_2 CONC.	SIMULIUM SP.
CHEMICAL OXYGEN DEMAND – $KMNO_4$	TUBIFEX SP.
CHEMICAL OXYGEN DEMAND – $K_2CR_2O_7$	
BIOLOGICAL OXYGEN DEMAND (BOD)	

rate of pollen and seed dispersal rate, which are the target attributes of this data set. The data have previously been analyzed in (Džeroski et al., 2005), where a detailed description can also be found.

SOLAR-FLARE

The same data set that we use as a multiple target classification problem can also be used as a multiple regression problem, since the target attributes are basically the numbers of solar flares, and can be regarded as numeric attributes. For more details on this data set, see Section 5.1.3.

WATER-QUALITY

This data set concerns the water quality of the Slovenian rivers. The data set comprises biological and chemical data that were collected through regular monitoring of rivers in Slovenia. The data source is the *Environmental Agency of the Republic of Slovenia* that performs water quality monitoring for most Slovenian rivers and maintains a database of water quality samples. The data cover a six year period, from 1990 to 1995, and have been previously analyzed by Džeroski et al. (2000). Biological samples are taken twice a year, once in summer and once in winter, while physical and chemical analyses are performed several times a year for each sampling site. The physical and chemical samples include the measured values of 16 different parameters. The biological samples include a list of all taxa (plant and animal species) present at the sampling site, together with their abundances. All the attributes of the data set are listed in Table 5.6. In total, 1060 water samples are available in the data set. In our experiments we have considered the physical and chemical properties as independent attributes, and the abundance of taxa as target attributes.

5.2 Comparison to existing methods

First, we compare the performance of predictive clustering rules to some existing methods. There are many rule learners for single target classification; we selected the CN2 rule learner (Clark and Niblett, 1989; Clark and Boswell, 1991) and a modification of CN2, the CN2-WRAcc (Todorovski et al., 2000), because our approach is a generalization of these algorithms. Additionally, we compare predictive clustering rules to two modern classification rule learners: Ripper (Cohen, 1995) and CN2-EVC (Možina et al., 2006). The latter one is, as the name implies, also an upgrade of the CN2 method. Rule learners for regression[3] are scarce; in fact, we were unable to find any working version of a regression rule learner, except for the FRS system (Demšar, 1999), which is a reimplementation of the FORS system (Karalič and Bratko, 1997). Unfortunately, this system cannot handle missing values, and we were only able to use it on a subset of the problems from Table 5.3. There are no existing rule learners for multiple target prediction, so we compare our system on these problems to predictive clustering trees (Blockeel et al., 1998) as implemented in the *Clus* system (Blockeel and Struyf, 2002). For completeness, we also compare predictive clustering rules to predictive clustering trees on single target problems. The overall significance of differences between a pair of methods is estimated using the *Wilcoxon signed-rank* test (Wilcoxon, 1945); in single target domains, each data set is a data point, while in multiple target domains, each target attribute of each data set corresponds to one data point. Comparison of many algorithms based on pair-wise comparisons can be difficult to follow (and is also theoreticaly flawed). For this reason we additionally perform the Nemenyi (Nemenyi, 1963) test on tasks where we compare more than three algorithms, and the results are visualised using the *average rank diagrams*

[3]Here we do not consider algorithms that generate rules via regression trees, e.g., *M5 Rules* (Holmes et al., 1999), or algorithms that use discretization to transform regression problems to classification problems, e.g., *SWAP1R* (Weiss and Indurkhya, 1993).

Table 5.7: *Single target classification*, comparison of *error rates* of *CN2*, *CN2-WRAcc*, *CN2-EVC*, *JRip* and *PCR* algorithms. *Significances* (p-values) of differences in error rates and rule set sizes for the pairs of algorithms. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller error rate or size. Significant differences are typeset in bold.

Compared algorithms		error p-value	size p-value
CN2 ordered	PCR ordered	0.978 <	0.151 >
CN2WRAcc ordered	PCR ordered	0.359 >	**0.003 <**
JRip ordered	PCR ordered	0.073 <	0.934 >
CN2EVC unordered	PCR ordered	0.421 <	**<0.001 >**
CN2 unordered	PCR unordered	**0.002 >**	0.804 <
CN2WRAcc unordered	PCR unordered	**0.003 >**	**<0.001 <**
JRip ordered	PCR unordered	0.847 >	**0.007 <**
CN2EVC unordered	PCR unordered	0.978 <	**0.007 >**
CN2 ordered	CN2 unordered	0.144 <	0.359 <
CN2WRAcc ordered	CN2WRAcc unordered	0.524 <	0.804 >
PCR ordered	PCR unordered	**0.018 >**	**<0.001 <**

(Demšar, 2006). When talking about significant differences we assume the standard *p-value* threshold of 0.05, however, one should not forget that significance is a continuous quantity.

5.2.1 CN2, CN2-WRAcc, CN2-EVC, and Ripper

The CN2 and CN2-WRAcc algorithms can induce ordered or unordered rules. CN2 can use significance testing for rule pruning, while there is no need for significance testing in CN2-WRAcc, since the number of induced rules by this algorithm is already much smaller (see Section 2.4 for more details). We use the *p-value* of 0.99 for significance testing in the CN2 algorithm. The CN2-EVC algorithm can only induce unordered rules and is implemented in the *Orange* data mining suite (Demšar et al., 2004). The Ripper algorithm can originally learn ordered or unordered rules, however, we used the *JRip* implementation from the *Weka* data mining suite (Witten and Frank, 2005) which can only learn ordered rules. Parameters of all algorithms were set to their default values.

Table 5.7 presents the results of significance testing; algorithms were compared pairwise, for both ordered rules and unordered rules. For each of the three algorithms that can induce ordered and unordered rules we also compared ordered vs. unordered rules. The error rates and rule set sizes for ordered and unordered rules are presented

Table 5.8: *Single target classification*, comparison of *error rates* of *CN2*, *CN2-WRAcc*, *JRip*, and *PCR* algorithms for *ordered rules*. For each data set, the smallest error rate is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

Data set	CN2			CN2-WRAcc			JRip			PCR		
	% error		# size	% error		# size	% error		# size	% error		# size
australian	21.6	±6.1	17	15.2	±3.5	3	**14.4**	±4.1	5	16.8	±6.6	14
balance	21.8	±4.8	31	27.7	±5.8	6	19.7	±5.3	12	**17.3**	±4.3	36
breast-w	5.1	±2.7	13	6.3	±3.1	11	5.6	±3.6	13	**4.6**	±2.8	5
bridges-td	**21.6**	±12.2	2	21.6	±13.6	3	22.6	±12.6	2	25.5	±15.8	4
car	4.6	±2.1	37	18.9	±4.6	11	12.6	±3.7	50	**3.5**	±1.3	40
glass	39.8	±13.1	10	38.8	±17.5	4	**30.4**	±12.7	9	37.4	±14.1	13
heart	24.1	±11.5	12	21.8	±9.0	3	**18.9**	±10.2	3	22.2	±7.8	11
image	**4.5**	±1.3	28	10.0	±2.1	8	4.7	±3.8	22	17.4	±1.7	11
iris	7.3	±6.6	2	8.0	±7.6	3	5.3	±7.6	4	**4.7**	±5.5	4
sonar	**23.4**	±14.7	10	28.8	±9.3	4	26.9	±9.1	5	32.7	±6.5	5
soya	13.5	±5.9	22	52.0	±8.4	8	**8.4**	±4.5	28	28.7	±14.7	15
tic-tac-toe	5.9	±6.6	32	30.3	±6.8	5	2.2	±3.4	9	**4.3**	±3.2	18
vote	7.1	±2.6	7	4.8	±3.0	5	**4.6**	±4.3	4	5.7	±4.0	7
waveform	21.9	±1.7	182	24.2	±2.4	6	19.9	±2.4	32	**19.7**	±1.9	16
wine	5.0	±4.9	3	**3.4**	±5.4	3	8.4	±6.0	3	9.0	±6.6	3
Average	15.2		27.2	20.8		5.5	13.6		13.4	16.7		13.5

in Table 5.8 and 5.9 respectively.

For ordered rules, we can see that there are no significant differences between the CN2, CN2-WRAcc, and PCR algorithms in terms or error, but rule sets induced by CN2-WRAcc have a significantly smaller number of rules. JRip is better than PCR in terms of error (*p-value*=0.073). Next, if we compare unordered rules, we see that predictive clustering rules have a significantly smaller error than CN2 and CN2-WRAcc algorithms. However, the PCR rule sets are much larger than the CN2-WRAcc rule sets. Even larger are the CN2-EVC rule sets. Finally, if we compare ordered and unordered rules induced by CN2, CN2-WRAcc, and PCR, the only significant difference is in the case of PCRs; unordered PCRs have a significantly smaller error, but this accuracy comes at a price, since their size is much larger.

The overall comparison of all mentioned algorithms (including predictive clustering trees, comparison to which is presented in Section 5.2.3) can be visually presented using the *average ranks diagrams* (Demšar, 2006). The diagrams comparing error rates and rule set sizes are given in Figures 5.1 and 5.2 respectively. The ruler in the diagram is the axis on which we plot the average ranks of compared methods. The methods with lower (i.e., better) ranks are on the right side of the diagram. The critical dif-

Table 5.9: *Single target classification*, comparison of *error rates* of *CN2, CN2-WRAcc, CN2-EVC* and *PCR* algorithms for *unordered rules*. For each data set, the smallest error rate is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

Data set	CN2 % error		CN2 # size	CN2-WRAcc % error		CN2-WRAcc # size	CN2-EVC % error		CN2-EVC # size	PCR % error		PCR # size
australian	16.1	±5.2	12	15.1	±3.6	2	14.2	±5.1	66	**13.6**	±5.1	26
balance	23.2	±5.4	51	30.3	±7.9	7	**16.8**	±5.2	68	17.4	±4.2	74
breast-w	7.0	±3.2	14	11.0	±5.1	5	20.0	±6.8	50	**3.6**	±1.8	10
bridges-td	**15.8**	±11.8	3	19.6	±10.4	3	16.5	±12.	5	17.6	±11.5	8
car	8.3	±2.5	96	30.0	±3.0	7	26.0	±3.4	87	**3.7**	±1.7	68
glass	42.4	±13.0	19	37.4	±18.1	7	**36.0**	±16.	36	38.8	±14.7	22
heart	27.4	±9.4	8	22.2	±7.2	3	**16.7**	±6.4	48	20.4	±9.3	17
image	13.7	±3.7	34	9.0	±2.6	7	19.2	±10.	72	**8.9**	±2.8	21
iris	11.4	±9.5	4	8.0	±7.6	3	**6.0**	±4.9	9	7.3	±6.6	5
sonar	37.0	±9.2	9	33.1	±7.8	5	**19.7**	±9.6	24	24.5	±12.1	8
soya	**12.2**	±4.0	37	47.7	±7.3	18	31.3	±7.8	50	13.6	±4.7	35
tic-tac-toe	3.2	±5.2	22	25.3	±6.5	7	**1.8**	±1.5	28	4.0	±3.6	52
vote	6.0	±3.1	8	4.4	±3.0	2	5.1	±3.6	36	**3.9**	±3.5	11
waveform	31.2	±2.9	78	27.1	±1.6	3	**17.8**	±1.5	308	18.9	±2.4	24
wine	7.3	±7.6	6	5.7	±5.3	4	**5.2**	±5.8	10	7.3	±5.3	6
Average	17.5		26.7	21.7		5.5	16.8		59.8	13.6		25.8

ference (CD) interval as computed by the Nemenyi test (Nemenyi, 1963) is plotted in the upper left corner; algorithms whose average ranks difference is larger than this critical difference can be considered significantly different with 95% probability. Algorithms that do not differ significantly are connected by a thick horizontal line. From the diagram comparing error rates (Figure 5.1) we can see that the algorithms can be roughly grouped in three groups: unordered PCRs, JRip, and CN2-EVC are ranked best, unordered CN2 and CN2-WRAcc are ranked worst, while the other three algorithms are in the middle. However, only the difference between the unordered PCRs and unordered CN2 is strictly significant (this suggests that the Nemenyi test is less powerful than the Wilcoxon signed-rank test which detects three significant differences). The algorithms on the diagram comparing rule set sizes (Figure 5.2) can also be grouped in three groups: CN2-WRAcc rule sets are ranked smallest, CN2-EVC and PC trees are ranked largest, while all other methods are in between. Taking both diagrams into account we can see that some methods trade accuracy for simplicity[4]

[4]The fact that gain in accuracy usually comes at a price of losing model simplicity is well known from decision trees (Bohanec and Bratko, 1994).

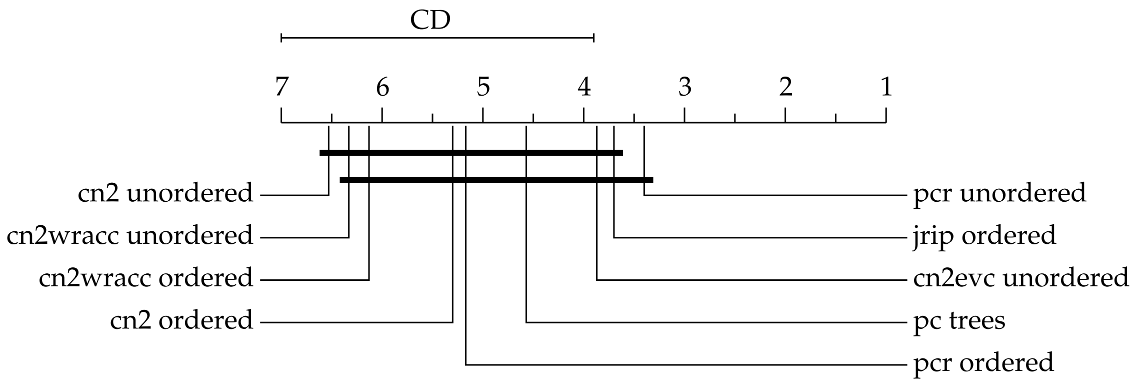


Figure 5.1: Average ranks diagram for *single target classification,* comparison of *error rates* of all algorithms compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

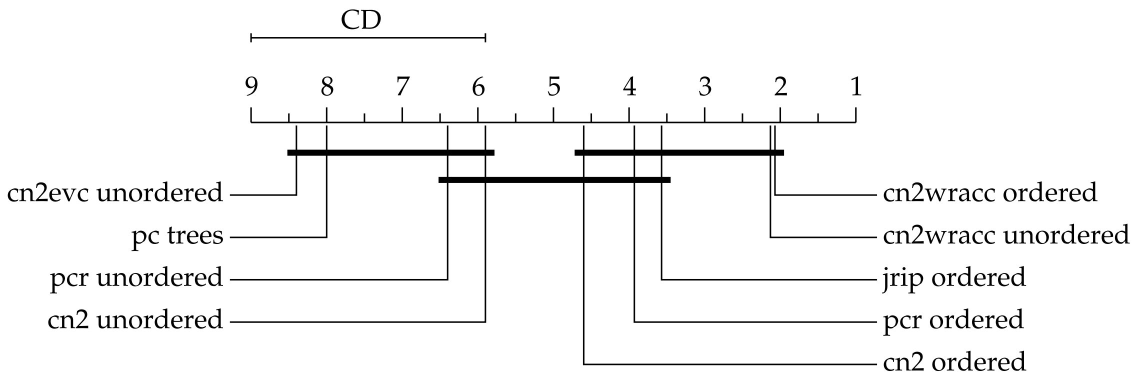


Figure 5.2: Average ranks diagram for *single target classification,* comparison of *rule set sizes* of all algorithms compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

(e.g., CN2-WRAcc) and vice versa (e.g., CN2-EVC) while others represent a compromise (e.g., JRip). A good rank of the JRip algorithm on both criteria can be explained with the fact that it includes the reduced error pruning algorithm, which the other algorithms do not. The fact that unordered PCRs are ranked first in terms of error rates, but only sixth in terms of rule sets sizes suggest that here also, some simplicity is traded for accuracy. A different parameter setting could change the relation between accuracy and simplicity. Ordered PCRs, however, seem to be a compromise in this regard.

From these results, we can conclude that the performance of predictive clustering rules on single target problems is comparable to the performance of the other algorithms for ordered rules, and comparable or better for unordered rules.

Table 5.10: *Single target regression*, comparison of *RRMSE* and *correlation coefficients* of *FRS* and *PCR* algorithms. *Significances* (p-values) of differences in RRMSE, correlation coefficients and rule set sizes for different algorithms. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller RRMSE, correlation coefficients or size. Significant differences are typeset in bold.

Compared algorithms		RRMSE p-value	CORR p-value	SIZE p-value
FRS ORDERED	PCR ORDERED	0.359 >	**0.020 <**	1.000 =
FRS ORDERED	PCR UNORDERED	0.129 >	**0.012 <**	0.496 >
PCR ORDERED	PCR UNORDERED	0.301 >	0.910 <	**0.004 >**

5.2.2 FRS

The FRS system (Demšar, 1999) is a reimplementation of the FORS system (Karalič and Bratko, 1997). It can learn ordered regression rules. The prediction model in each rule can either be a constant or a linear regression model. In order to compare FRS to predictive clustering rules (PCRs), we have disabled the construction of linear regression models, and set the minimal number of examples covered by each rule to be the same in both algorithms (i.e., 2). All the other FRS settings were set to their default values. Unfortunately, the FRS system cannot handle missing values, and we were only able to use it on a subset of the problems from Table 5.3. At this point we should note that the FRS was designed as an inductive logic programming system for linear regression and we use it here for a task for which it was not originally intended. However, since this was the only regression rule learner we were able to find, and it uses a similar rule learning as CN2, we believe such a comparison is beneficial.

The significances of differences between the FRS and PCR algorithms are presented in Table 5.10; the relative root mean squared errors (RRMSE), correlation coefficients, and rule set sizes are presented in Table 5.11 and 5.12, respectively. When looking at the RRMSE, we can see that ordered and unordered PCRs perform better than rules generated by FRS, but this difference is not significant. In the correlation coefficient comparison, the correlation coefficients of PCRs are larger, and this time the difference is significant.

There is virtually no difference in the rule set sizes between FRS and the ordered PCR rule sets; the unordered PCR rule sets are smaller, the difference being almost significant (*p-value*=0.129). The differences between ordered and unordered PCRs are not significant, except for the rule set size: unordered PCR rule sets are significantly smaller. Based on these results, we can conclude that PCRs are comparable or better than the FRS rules.

Table 5.11: *Single target regression,* comparison of *RRMSE* of *FRS* and *PCR* algorithms. For each data set, the smallest RRMSE is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

Data set	frs ordered		pcr ordered		pcr unordered	
	rrmse	# size	rrmse	# size	rrmse	# size
auto-price	0.91 ±0.01	19	**0.52** ±0.13	3	**0.52** ±0.15	3
cloud	0.94 ±0.03	1	**0.72** ±0.24	3	0.81 ±0.25	3
cpu	0.88 ±0.01	18	**0.52** ±0.29	3	0.65 ±0.42	3
housing	0.80 ±0.00	126	**0.67** ±0.12	7	**0.67** ±0.11	7
quake	**1.00** ±0.00	4	1.16 ±0.05	129	1.02 ±0.04	8
sensory	**1.00** ±0.00	1	1.07 ±0.10	34	**1.00** ±0.10	10
servo	1.01 ±0.02	1	0.61 ±0.32	5	**0.49** ±0.20	3
strike	**0.99** ±0.01	19	1.26 ±0.48	14	1.10 ±0.41	13
veteran	**1.02** ±0.04	1	1.22 ±0.48	8	1.09 ±0.27	5
Average	0.95	21.1	0.86	22.9	0.82	6.1

Table 5.12: *Single target regression,* comparison of *correlation coefficients* of *FRS* and *PCR* algorithms. For each data set, the largest correlation coefficient is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

Data set	frs ordered		pcr ordered		pcr unordered	
	corr	# size	corr	# size	corr	# size
auto-price	0.41 ±0.04	19	**0.86** ±0.07	3	**0.86** ±0.08	3
cloud	0.35 ±0.10	1	**0.71** ±0.28	3	0.64 ±0.19	3
cpu	0.50 ±0.22	18	**0.86** ±0.31	3	0.76 ±0.29	3
housing	0.62 ±0.03	126	**0.76** ±0.09	7	**0.76** ±0.08	7
quake	**0.09** ±0.02	4	0.01 ±0.06	129	0.02 ±0.05	8
sensory	-0.08 ±0.06	1	**0.27** ±0.18	34	0.22 ±0.19	10
servo	-0.22 ±0.21	1	0.82 ±0.22	5	**0.87** ±0.16	3
strike	0.17 ±0.03	19	0.14 ±0.24	14	**0.19** ±0.22	13
veteran	-0.03 ±0.07	1	0.10 ±0.31	8	**0.19** ±0.36	5
Average	0.20	21.1	0.50	22.9	0.50	6.1

Table 5.13: *Single target classification,* comparison of *error rates* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered* and *unordered rules.* *Significances* (p-values) of differences in error rates and rule set sizes for the pairs of algorithms. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller error rate or size. Significant differences are typeset in bold.

Compared algorithms		error p-value	size p-value
PC TREES	PCR ORDERED	0.679 <	**<0.001 >**
PC TREES	PCR UNORDERED	0.303 >	**<0.001 >**

5.2.3 Predictive clustering trees

Predictive clustering trees (PCTs) are the original predictive clustering method introduced by Blockeel (1998); Blockeel et al. (1998). We have used PCTs as implemented in the *Clus* machine learning toolkit (Blockeel and Struyf, 2002). Where applicable, both the PCT algorithm and the PCR algorithm used the same settings, while all tree-specific settings were set to their defaults. PCTs can be used to learn models on the same types of tasks as PCRs can, i.e., they can learn models in single and multiple target, classification and regression problem domains. We present the results for each problem domain type separately.

Trees on single target classification problems

The significances of differences between predictive clustering trees (PCTs) and PCRs are given in Table 5.13, while the error rates are presented in Table 5.14. Trees have a slightly lower error rate than ordered rules (*p-value*<0.679), and a slightly higher error rate than unordered rules (*p-value*<0.303). The differences, however, are not significant. On the other hand, the number of rules produced in each case is significantly smaller (*p-value*<0.001) than the number of leaves in the trees (i.e., the number of rules produced by the conversion of trees to rules). The comparison of PCTs to all other rule learning algorithms in terms of average ranks is given in Figures 5.1 and 5.2.

Trees on single target regression problems

The significances of pairwise differences between predictive clustering trees (PCTs), and ordered and unordered PCR rule sets are presented in Table 5.15; the relative root mean squared errors (RRMSE), correlation coefficients, and rule set sizes are given in Tables 5.16 and 5.17, respectively. Ordered rules are comparable to trees in size, but much worse (*p-value*<0.001) in terms of RRMSE, as well as in terms of

Table 5.14: *Single target classification,* comparison of *error rates* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered* and *unordered rules.* For each data set, the smallest error rate is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The last row gives the averages over all data sets.

Data set	pc trees			pcr ordered			pcr unordered		
	% error		# size	% error		# size	% error		# size
australian	19.4	±5.6	62	16.8	±6.6	14	**13.6**	±5.1	26
balance	21.6	±4.7	101	**17.3**	±4.3	36	17.4	±4.2	74
breast-w	6.3	±2.4	19	4.6	±2.8	5	**3.6**	±1.8	10
bridges-td	**14.7**	±8.4	4	25.5	±15.8	4	17.6	±11.5	8
car	**2.1**	±1.3	67	3.5	±1.3	40	3.7	±1.7	68
glass	**29.4**	±13.0	34	37.4	±14.1	13	38.8	±14.7	22
heart	27.0	±9.2	31	22.2	±7.8	11	**20.4**	±9.3	17
image	**4.2**	±1.5	49	17.4	±1.7	11	8.9	±2.8	21
iris	6.7	±4.4	5	**4.7**	±5.5	4	7.3	±6.6	5
sonar	26.9	±6.7	19	32.7	±6.5	5	**24.5**	±12.1	8
soya	**13.0**	±6.1	56	28.7	±14.7	15	13.6	±4.7	35
tic-tac-toe	7.4	±3.7	58	4.3	±3.2	18	**4.0**	±3.6	52
vote	5.1	±2.4	17	5.7	±4.0	7	**3.9**	±3.5	11
waveform	23.4	±1.6	421	19.7	±1.9	16	**18.9**	±2.4	24
wine	10.7	±9.9	8	9.0	±6.6	3	**7.3**	±5.3	6
Average	14.5		63.4	16.6		13.5	13.6		25.8

Table 5.15: *Single target regression,* comparison of *RRMSE* and *correlation coefficients* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered* and *unordered rules. Significances* (p-values) of differences in RRMSE, correlation coefficients and rule set sizes for the pairs of algorithms. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller RRMSE, correlation coefficients or size. Significant differences are typeset in bold.

Compared algorithms		rrms p-value	corr p-value	size p-value
pc trees	pcr ordered	**<0.001 <**	**<0.001 >**	0.978 >
pc trees	pcr unordered	**<0.001 <**	**<0.001 >**	0.095 >
pcr ordered	pcr unordered	**0.048 >**	0.762 <	**0.015 >**

Table 5.16: *Single target regression,* comparison of *RRMSE* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered* and *unordered rules.* For each data set, the smallest RRMSE is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

DATA SET	PC TREES		PCR ORDERED		PCR UNORDERED	
	RRMSE	# SIZE	RRMSE	# SIZE	RRMSE	# SIZE
AUTO-HORSE	**0.41** ±0.17	23	0.68 ±0.18	4	0.63 ±0.16	4
AUTO-MPG	**0.45** ±0.08	16	0.70 ±0.10	4	0.66 ±0.11	4
AUTO-PRICE	**0.49** ±0.15	9	0.52 ±0.13	3	0.52 ±0.15	3
BREAST-TUMOR	**0.96** ±0.13	3	1.12 ±0.19	17	1.03 ±0.15	9
CLOUD	**0.58** ±0.24	9	0.72 ±0.24	3	0.81 ±0.25	3
CPU	**0.33** ±0.17	12	0.52 ±0.29	3	0.65 ±0.42	3
ECHO-MONTHS	**0.68** ±0.19	3	0.85 ±0.32	3	0.81 ±0.29	3
HOUSING	**0.43** ±0.08	32	0.67 ±0.12	7	0.67 ±0.11	7
META	**1.03** ±0.83	0	1.16 ±0.99	1	1.15 ±1.10	1
PBC	**0.94** ±0.15	6	1.01 ±0.18	18	**0.94** ±0.15	4
QUAKE	**0.99** ±0.04	3	1.16 ±0.05	129	1.02 ±0.04	8
SENSORY	**0.94** ±0.07	7	1.07 ±0.10	34	1.00 ±0.10	10
SERVO	**0.42** ±0.19	11	0.61 ±0.32	5	0.49 ±0.20	3
STRIKE	**0.98** ±0.44	12	1.26 ±0.48	14	1.10 ±0.41	13
VETERAN	**0.99** ±0.37	2	1.22 ±0.48	8	1.09 ±0.27	5
AVERAGE	0.71	9.9	0.88	16.9	0.84	5.3

correlation coefficient. Unordered rules are also much worse than trees in terms of RRMSE and correlation coefficient (*p-value*<0.001), but the rule sets are smaller than trees (*p-value*<0.095). Comparison between ordered and unordered rules shows that unordered rules are better in terms of RRMSE (*p-value*<0.048), while there are no significant differences in terms of correlation coefficient (*p-value*<0.762); unordered rule sets are smaller than ordered rule sets (*p-value*<0.015). Possible reasons for the worse performance of rules in comparison to trees are discussed in Section 5.2.4.

Trees on multiple target classification problems

The significances of differences between predictive clustering trees (PCTs) and PCRs on multiple target classification domains are given in Table 5.18, while the error rates are presented in Table 5.19. The results show that, in most cases, one method has a lower error rate on all or most of the target attributes within each data set, e.g., unordered PCRs are better on the BRIDGES, MEDIANA, and WATER-QUALITY-DIS data sets, while PCTs are better on the EDM-DIS, MONKS, SOLAR-FLARE-DIS, and THYROID-0387 data sets. The overall comparison (Table 5.18) shows that PCTs have comparable

Table 5.17: *Single target regression,* comparison of *correlation coefficients* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered* and *unordered rules.* For each data set, the largest correlation coefficient is typeset in bold. Size is given as the number of learned rules. The last row gives the averages over all data sets.

Data set	PC trees		PCR ordered		PCR unordered	
	corr	# size	corr	# size	corr	# size
auto-horse	**0.91** ±0.21	23	0.74 ±0.19	4	0.78 ±0.16	4
auto-mpg	**0.89** ±0.03	16	0.73 ±0.07	4	0.76 ±0.07	4
auto-price	**0.88** ±0.05	9	0.86 ±0.07	3	0.86 ±0.08	3
breast-tumor	**0.29** ±0.17	3	0.20 ±0.23	17	0.12 ±0.17	9
cloud	**0.82** ±0.10	9	0.71 ±0.28	3	0.64 ±0.19	3
cpu	**0.94** ±0.06	12	0.86 ±0.31	3	0.76 ±0.29	3
echo-months	**0.74** ±0.22	3	0.57 ±0.32	3	0.60 ±0.32	3
housing	**0.90** ±0.04	32	0.76 ±0.09	7	0.76 ±0.08	7
meta	0.04 ±0.33	0	0.07 ±0.35	1	**0.05** ±0.37	1
pbc	0.39 ±0.20	6	0.37 ±0.13	18	**0.40** ±0.17	4
quake	**0.12** ±0.10	3	0.01 ±0.06	129	0.02 ±0.05	8
sensory	**0.37** ±0.06	7	0.27 ±0.18	34	0.22 ±0.19	10
servo	**0.91** ±0.06	11	0.82 ±0.22	5	0.87 ±0.16	3
strike	**0.33** ±0.24	12	0.14 ±0.24	14	0.19 ±0.22	13
veteran	**0.24** ±0.34	2	0.10 ±0.31	8	0.19 ±0.36	5
Average	0.58	9.9	0.48	16.9	0.48	5.3

error rates to ordered PCRs. Unordered PCRs are somewhat better than trees, but the difference is only weakly significant (*p-value*=0.13). Unordered PCRs are, however, significantly better than ordered PCRs (*p-value*=0.014).

When considering the rule set sizes (trees are converted to rules), we see that ordered PCR rule sets are significantly smaller than unordered PCRs and PCTs (*p-value*<0.001), and unordered PCR rule sets are in turn significantly smaller than rule sets obtained from trees (*p-value*<0.001). The results suggest that ordered PCRs tend to produce smaller, while unordered PCRs produce more accurate rule sets, compared to the other two methods. These results are similar as in the case of single target classification.

Trees on multiple target regression problems

The significances of differences between predictive clustering trees (PCTs), as well as ordered and unordered PCRs on multiple target regression domains are given in Table 5.20. The relative root mean squared errors (RRMSE), correlation coefficients, and rule set sizes are presented in Tables 5.21 and 5.22 respectively. The results from

Table 5.18: *Multiple target classification,* comparison of *error rates* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered and unordered rules. Significances* (p-values) of differences in error rates and rule set sizes for the pairs of algorithms over all data sets and all target attributes. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller error rate or size. Significant differences are typeset in bold.

Compared algorithms		error p-value	size p-value
pc trees	pcr ordered	0.819 <	**<0.001 >**
pc trees	pcr unordered	0.130 >	**<0.001 >**
pcr ordered	pcr unordered	**0.014 >**	**<0.001 <**

the first table, i.e., the overall comparison of differences across all target attributes and all data sets suggest that PCRs are much worse than the trees, since the RRMSE and correlation coefficients are significantly worse (in all cases *p-value*<0.003). However, if we take a look at the results for each data set separately (Tables 5.21 and 5.22), we can see that there exist data sets (e.g., the microarthropods and sigmea-real) on which ordered or unordered PCRs are comparable to or even slightly better than the trees. When comparing ordered and unordered rules, unordered are better in terms of RRMSE (*p-value*=0.104) and worse in terms of correlation coefficients (*p-value*=0.388), which suggests that there are no significant differences between the two algorithms in terms of accuracy. This results are somewhat different from the single target regression case, where unordered rules were better.

When comparing the rule set sizes, ordered PCR rule sets are significantly larger than rule sets obtained from trees. On the other hand, unordered PCR rule sets are a smaller than trees (*p-value*=0.238) and much smaller than ordered PCRs (*p-value*<0.001). This results are in accordance with the results on single target regression problems.

Table 5.19: *Multiple target classification,* comparison of *error rates* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered and unordered rules.* For each data set, the average error rate over all target attributes is given first, and then for each target attribute separately. In each row, the smallest error rate is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row (next page) gives the average error rate over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PC TREES			PCR ORDERED			PCR UNORDERED		
TAR. ATT.	% ERROR		# SIZE	% ERROR		# SIZE	% ERROR		# SIZE
BRIDGES	34.8		28	40.5		7	**32.2**		12
T-OR-D	14.1	±7.3		24.7	±0.0		**10.6**	±8.7	
MATERIAL	27.1	±0.0		20.0	±10.1		**18.8**	±10.4	
SPAN	**35.3**	±15.2		43.5	±11.3		40.0	±11.0	
REL-L	41.2	±0.0		44.7	±0.0		**35.3**	±15.7	
TYPE	**56.5**	±0.0		69.4	±0.0		**56.5**	±0.0	
EDM-DIS	**22.8**		22	25.0		9	29.2		11
D-FLOW	**11.7**	±6.0		**11.7**	±8.1		12.3	±9.0	
D-GAP	**33.8**	±15.9		38.3	±8.2		46.1	±13.4	
MEDIANA	19.6		1948	17.2		271	**16.6**		685
READ-DELO	26.2	±1.7		22.0	±1.5		**21.7**	±1.2	
READ-DNEVNIK	19.5	±1.6		16.6	±1.4		**15.4**	±0.9	
READ-EKIPA	8.3	±0.7		7.1	±0.8		**6.3**	±0.8	
READ-SL-NOV	32.0	±1.3		**28.8**	±1.1		29.2	±0.9	
READ-VECER	11.8	±1.3		11.6	±0.9		**10.4**	±1.0	
MONKS	**17.2**		68	21.7		4	23.5		10
MONK-1	**9.3**	±7.7		30.1	±9.1		17.6	±7.3	
MONK-2	41.9	±11.6		**33.1**	±8.0		35.9	±5.9	
MONK-3	**0.5**	±1.5		1.9	±3.0		17.1	±5.4	
SIGMEA-REAL-DIS	29.4		187	**24.9**		38	**24.9**		72
MFO	29.9	±6.6		**24.5**	±5.2		25.1	±4.6	
MSO	28.9	±4.6		25.3	±3.8		**24.6**	±3.3	
SIGMEA-SIM-DIS	**0.1**		35	2.1		3	2.1		4
DISP-RATE	**0.3**	±0.2		4.3	±0.7		4.3	±0.7	
DISP-SEEDS	**0.0**	±0.0		**0.0**	±0.0		**0.0**	±0.0	
SOLAR-FLARE-DIS	**9.6**		26	11.0		23	10.4		39
C-CLASS	**13.3**	±9.5		15.2	±6.7		13.6	±6.9	
M-CLASS	**13.3**	±4.8		14.6	±4.7		14.9	±4.6	
X-CLASS	**2.2**	±3.2		3.1	±3.2		2.8	±3.4	

Continued on the next page.

Table 5.19: Continued from the previous page.

Data set tar. att.	pc trees % error	 # size	pcr ordered % error	 # size	pcr unordered % error	 # size
thyroid-0387	**1.0**	298	2.4	497	2.5	560
hyper-thyro	**1.1** ±0.3		2.5 ±0.6		2.5 ±0.5	
hypo-thyro	**0.8** ±0.3		3.1 ±0.8		3.7 ±0.5	
bind-prot	**1.6** ±0.4		3.2 ±0.8		3.4 ±0.6	
gen-health	**1.0** ±0.4		3.6 ±0.6		2.7 ±0.9	
repl-theory	**1.0** ±0.3		2.1 ±0.4		3.0 ±0.8	
antithyro-tr	**0.3** ±0.1		**0.3** ±0.2		0.4 ±0.2	
disc-results	**1.0** ±0.2		2.0 ±0.5		2.0 ±0.6	
water-quality-dis	34.3	444	33.3	89	**31.8**	153
clad-sp	40.8 ±3.4		**39.5** ±5.6		40.4 ±4.9	
gong-inc	33.6 ±3.4		29.5 ±3.6		**28.4** ±3.2	
oedo-sp	30.2 ±4.2		**29.7** ±4.5		29.9 ±5.1	
tige-ten	23.9 ±4.2		23.2 ±4.6		**20.8** ±2.4	
melo-var	43.2 ±7.6		41.7 ±4.8		**41.4** ±3.7	
nitz-pal	35.4 ±6.6		31.3 ±2.3		**30.6** ±4.3	
audo-cha	30.2 ±6.4		29.3 ±5.0		**24.2** ±5.2	
erpo-oct	31.1 ±3.7		29.0 ±2.6		**26.5** ±3.3	
gamm-foss	**37.8** ±4.5		38.1 ±4.5		**37.8** ±3.7	
baet-rhod	32.4 ±5.3		**31.8** ±3.1		32.5 ±2.4	
hydro-sp	39.5 ±5.2		39.1 ±4.6		**38.2** ±4.9	
rhya-sp	31.6 ±3.6		36.1 ±5.5		**30.8** ±6.8	
simu-sp	40.6 ±5.1		38.5 ±5.5		**37.3** ±4.7	
tubi-sp	29.8 ±3.4		28.8 ±4.0		**27.1** ±3.1	
Average	22.0	339.6	22.6	104.6	**21.4**	171.8

Table 5.20: *Multiple target regression,* comparison of *RRMSE* and *correlation coefficients* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered and unordered rules. Significances* (p-values) of differences in RRMSE, correlation coefficients, and rule set sizes for the pairs of algorithms over all data sets and all target attributes. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller RRMSE, correlation coefficients or size. Significant differences are typeset in bold.

Compared algorithms		rrmse p-value	corr p-value	size p-value
pc trees	pcr ordered	**<0.001 <**	**0.003 >**	**<0.001 <**
pc trees	pcr unordered	**<0.001 <**	**<0.001 >**	0.238 <
pcr ordered	pcr unordered	0.104 >	0.388 >	**<0.001 >**

Table 5.21: *Multiple target regression,* comparison of *RRMSE* of *predictive clustering trees (PCTs),* and *PCR* algorithms for *ordered and unordered rules.* For each data set, the average RRMSE over all target attributes is given first, and then for each target attribute separately. In each row, the smallest RRMSE is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average RRMSE over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PC TREES		PCR ORDERED		PCR UNORDERED	
TAR. ATT.	RRMSE	# SIZE	RRMSE	# SIZE	RRMSE	# SIZE
EDM	**0.72**	11	0.88	9	0.92	4
D-FLOW	**0.69** ±0.42		0.94 ±0.43		1.04 ±0.37	
D-GAP	**0.76** ±0.10		0.82 ±0.09		0.80 ±0.09	
MICROARTHROPODS	0.87	50	**0.85**	108	0.96	7
ACARI	0.91 ±0.19		**0.89** ±0.17		0.99 ±0.22	
COLLEMBOLAN	0.92 ±0.20		**0.90** ±0.19		0.96 ±0.20	
SH-BIODIV	0.78 ±0.03		**0.76** ±0.04		0.93 ±0.06	
SIGMEA-REAL	0.61	7	0.61	9	**0.60**	7
MFO	**0.62** ±0.38		0.74 ±0.47		0.70 ±0.38	
MSO	0.61 ±0.44		**0.49** ±0.32		0.50 ±0.30	
SIGMEA-SIM	**0.03**	166	0.13	2	0.40	2
DISP-RATE	**0.03** ±0.01		0.15 ±0.00		0.44 ±0.00	
DISP-SEEDS	**0.03** ±0.01		0.11 ±0.00		0.35 ±0.00	
SOLAR-FLARE	**1.00**	2	1.15	13	1.10	12
C-CLASS	**0.99** ±0.31		1.04 ±0.30		1.02 ±0.31	
M-CLASS	**0.98** ±0.39		1.09 ±0.36		1.05 ±0.36	
X-CLASS	**1.02** ±0.73		1.32 ±0.77		1.24 ±0.74	
WATER-QUALITY	**0.96**	5	1.09	185	0.99	31
CLAD-SP	**0.99** ±0.06		1.10 ±0.10		**0.99** ±0.00	
GONG-INC	**0.99** ±0.09		1.14 ±0.11		1.01 ±0.10	
OEDO-SP	**0.99** ±0.11		1.17 ±0.18		**0.99** ±0.10	
TIGE-TEN	**0.93** ±0.14		1.08 ±0.10		0.97 ±0.16	
MELO-VAR	**0.98** ±0.08		1.09 ±0.05		1.00 ±0.08	
NITZ-PAL	**0.90** ±0.07		0.98 ±0.08		0.96 ±0.06	
AUDO-CHA	**0.98** ±0.00		1.19 ±0.00		1.00 ±0.13	
ERPO-OCT	**0.95** ±0.12		1.06 ±0.09		0.98 ±0.13	
GAMM-FOSS	**0.93** ±0.06		1.00 ±0.07		0.96 ±0.05	
BAET-RHOD	**0.98** ±0.12		1.05 ±0.00		0.99 ±0.13	
HYDRO-SP	**0.97** ±0.08		1.11 ±0.08		1.00 ±0.07	
RHYA-SP	**0.95** ±0.15		1.12 ±0.12		0.99 ±0.14	
SIMU-SP	**1.00** ±0.05		1.15 ±0.00		1.03 ±0.05	
TUBI-SP	**0.89** ±0.08		1.00 ±0.11		0.94 ±0.12	
AVERAGE	**0.84**	40.2	0.94	54.3	0.92	10.5

Table 5.22: *Multiple target regression*, comparison of *correlation coefficients* of *predictive clustering trees (PCTs)*, and *PCR* algorithms for *ordered and unordered rules*. For each data set, the average correlation coefficient over all target attributes is given first, and then for each target attribute separately. In each row, the largest correlation coefficient is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average correlation coefficient over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PC TREES		PCR ORDERED		PCR UNORDERED	
TAR. ATT.	CORR	# SIZE	CORR	# SIZE	CORR	# SIZE
EDM	**0.71**	11	0.60	9	0.49	4
D-FLOW	**0.75** ±0.49		0.61 ±0.39		0.39 ±0.32	
D-GAP	**0.66** ±0.12		0.59 ±0.13		0.60 ±0.13	
MICROARTHROPODS	0.48	50	**0.52**	108	0.27	7
ACARI	0.41 ±0.08		**0.47** ±0.09		0.17 ±0.08	
COLLEMBOLAN	0.40 ±0.05		**0.43** ±0.08		0.28 ±0.08	
SH-BIODIV	0.64 ±0.04		**0.66** ±0.06		0.37 ±0.07	
SIGMEA-REAL	0.79	7	0.79	9	**0.81**	7
MFO	**0.79** ±0.21		0.72 ±0.24		0.75 ±0.24	
MSO	0.79 ±0.25		**0.87** ±0.26		**0.87** ±0.25	
SIGMEA-SIM	**1.00**	166	0.99	2	0.95	2
DISP-RATE	**1.00** ±0.00		0.99 ±0.00		0.94 ±0.00	
DISP-SEEDS	**1.00** ±0.00		0.99 ±0.00		0.97 ±0.00	
SOLAR-FLARE	**0.13**	2	0.09	13	0.10	12
C-CLASS	0.16 ±0.25		0.17 ±0.31		**0.19** ±0.30	
M-CLASS	**0.21** ±0.30		0.12 ±0.19		0.15 ±0.16	
X-CLASS	**0.03** ±0.15		-0.03 ±0.03		-0.04 ±0.04	
WATER-QUALITY	**0.26**	5	0.19	185	0.19	31
CLAD-SP	0.14 ±0.11		0.13 ±0.09		**0.19** ±0.00	
GONG-INC	**0.12** ±0.06		-0.02 ±0.09		0.04 ±0.08	
OEDO-SP	0.12 ±0.07		0.12 ±0.11		**0.20** ±0.10	
TIGE-TEN	**0.37** ±0.09		0.27 ±0.12		0.26 ±0.16	
MELO-VAR	**0.21** ±0.14		0.19 ±0.12		0.13 ±0.07	
NITZ-PAL	**0.43** ±0.09		0.34 ±0.11		0.30 ±0.12	
AUDO-CHA	**0.21** ±0.00		0.11 ±0.00		0.12 ±0.08	
ERPO-OCT	**0.33** ±0.08		0.24 ±0.12		0.23 ±0.14	
GAMM-FOSS	**0.38** ±0.15		0.31 ±0.10		0.30 ±0.08	
BAET-RHOD	0.20 ±0.11		**0.21** ±0.00		0.18 ±0.10	
HYDRO-SP	**0.26** ±0.04		0.12 ±0.09		0.13 ±0.09	
RHYA-SP	**0.32** ±0.09		0.20 ±0.07		0.21 ±0.14	
SIMU-SP	**0.10** ±0.10		0.07 ±0.00		0.06 ±0.10	
TUBI-SP	**0.45** ±0.10		0.33 ±0.15		0.36 ±0.13	
AVERAGE	**0.40**	40.2	0.35	54.3	0.32	10.5

Table 5.23: An example of single target regression rules learned on the ECHO-MONTHS domain. Starting from the top we are given rules transcribed form the PCT trees, ordered PCR rules, and unordered PCR rules.

Rules from a tree

Rule 1:	`IF` $(still\_alive = 0)$ `THEN` $[survival = 29.9]$
Rule 2:	`IF` $(still\_alive = 1) \wedge (alive\_at\_1 = 0)$ `THEN` $[survival = 18.0]$
Rule 3:	`IF` $(still\_alive = 1) \wedge (alive\_at\_1 = 1)$ `THEN` $[survival = 3.3]$

RRMSE = 0.68, Corr. coeff. = 0.74

Ordered rules

Rule 1:	`IF` $(still\_alive = 0) \wedge (wall\_score \leq 21.5)$ `THEN` $[survival = 29.9]$
Rule 2:	`IF` $(still\_alive = 1)$ `THEN` $[survival = 5.9]$
Rule 3:	`IF` $(wall\_index \leq 2.2) \wedge (age > 55)$ `THEN` $[survival = 18.3]$
Otherwise:	$[survival = 54]$

RRMSE = 0.85, Corr. coeff. = 0.57

Unordered rules

Rule 1:	`IF` $(still\_alive = 0) \wedge (wall\_score \leq 21.5)$ `THEN` $[survival = 29.9]$
Rule 2:	`IF` $(still\_alive = 1)$ `THEN` $[survival = 5.9]$
Rule 3:	`IF` $(wall\_index \leq 2.2) \wedge (age > 55)$ `THEN` $[survival = 18.3]$
Otherwise:	$[survival = 54]$

RRMSE = 0.81, Corr. coeff. = 0.60

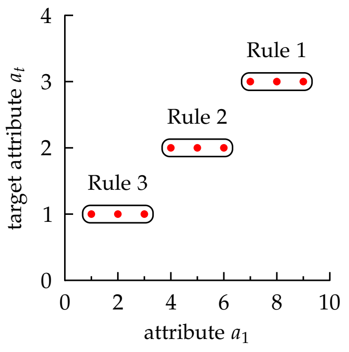
5.2.4 Discussion on regression rules

Previous sections (5.2.1, 5.2.2, and 5.2.3) presented the comparison of predictive clustering rules (PCR) to some existing methods. On the classification problems the results indicate that PCRs yield models of roughly comparable accuracy as predictive clustering trees (PCT). On the regression problems, however, PCRs are much worse than PCTs. On the other hand, comparison of PCRs to the regression rules learning method FRS shows no significant difference in accuracy. This can be illustrated by an example of ordered and unordered PCRs, as well as rules transcribed from a regression tree given in Table 5.23 (the examples were learned on a single target domain ECHO-MONTHS). We can see that all three rule sets have an equal number of rules, however the 'tree' rules are better than unordered PCRs, which are in turn slightly better than ordered PCRs. It is interesting that ordered and unordered PCR rule sets comprise exactly the same rules, however, unordered rules are slightly better because of the rule set interpretation by weighted voting. Unfortunately, these rule sets are less accurate than the rule set obtained from the tree. It seems that the PCT rule learning algorithm was unable to find the most discriminative attributes: the 'tree' rule set comprises attributes *'still_alive'* and *'alive_at_1'*, while the PCR rule sets include, besides *'still_alive'*, the attributes *'wall_score'*, *'wall_index'*, and *'age'*, which are obviously less appropriate.

We can think of two main reasons why PCTs are more accurate than PCRs. The first reason is that the regression PCTs use the tree post-pruning approach from the M5 algorithm (Quinlan, 1992). This method is known to improve the accuracy and understandability of M5 regression and model trees. The PCR algorithm, on the other hand does not use any form of rule post-pruning. We believe this is the main reason why PCTs are much better than PCRs. A remedy to this problem should be the inclusion of post-pruning in the PCR algorithm, e.g., the *reduced error pruning* (Brunk and Pazzani, 1991) as in the Ripper algorithm (Cohen, 1995).

An additional reason why the regression rules are worse than regression trees could be the fact that the covering algorithm with its *separate and conquer* strategy is less appropriate for learning of regression rules than the *divide and conquer* strategy used in the top down induction algorithm for learning regression trees. In the process of tree learning, the entire learning set is always considered, and a new split is added to the tree only if it improves the accuracy of a tree on a certain part of the example space, and as a consequence, also on the entire example space. On the other hand, when generating a new rule with the standard covering algorithm, only the examples that have not yet been covered are considered. This means that new rules are learned independently of the already learned rules. The new rule learned in this way may be optimal for the current learning set, but not for the entire learning set, and in addition, it might not be optimal when taking into account the previously learned rules. In the experiments performed in previous sections we tried to alleviate this problem by setting the *rule adding method* (M_{add}) of the PCR algorithm to *"If-Better"*, which means that a newly learned rule is added to the rule set only if it improves its overall performance (see Section 4.4).

In addition, we have noticed an interesting effect that the overlapping of rules,



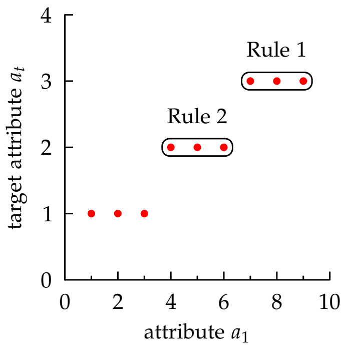
Rules from a tree

Rule 1: IF $(a_1 > 6)$ THEN $[a_t = 3]$

Rule 2: IF $(a_1 \leq 6) \wedge (a_1 > 3)$ THEN $[a_t = 2]$

Rule 3: IF $(a_1 \leq 3)$ THEN $[a_t = 1]$

Figure 5.3: Unordered regression rules transcribed from a regression tree learned on the artificial data SYNTH-1.



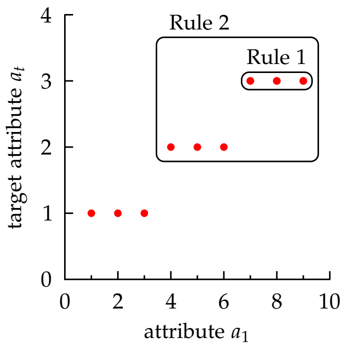
Ordered rules

Rule 1: IF $(a_1 > 6)$ THEN $[a_t = 3]$

Rule 2: IF $(a_1 > 3)$ THEN $[a_t = 2]$

Otherwise: $[a_t = 1]$

Figure 5.4: Ordered regression rules learned on the artificial data SYNTH-1.



Unordered rules

Rule 1: IF $(a_1 > 6)$ THEN $[a_t = 3]$

Rule 2: IF $(a_1 > 3)$ THEN $[a_t = 2]$

Otherwise: $[a_t = 1]$

Figure 5.5: Unordered regression rules learned on the artificial data SYNTH-1.

i.e., when one example is covered by more than one rule (this effect is only relevant for unordered rules), can decrease the overall accuracy. We illustrate the problem on a very simple artificial data set with only one descriptive and one target attribute. The data set (SYNTH-1) consists of 9 examples with different values of the descriptive attribute (a_1), and three different monotonically increasing values (1, 2, and 3) of the target attribute (a_t). On this data set we learn predictive clustering trees (PCTs), and ordered as well as unordered predictive clustering rules (PCRs). The resulting rule sets (tree was converted into rules) are presented in Figures 5.3, 5.4, and 5.5, respectively. In each figure we have a plot representing the example space with the descriptive attribute values on the x-axis and the target attribute values on the y-axis; each learning example is depicted as a dot. On the right of the plot we have the rules that were learned on this data set.

Let us first look at the rules which correspond to a tree (Figure 5.3). Such rules are unordered and all together cover the entire example space; there is no need for a default rule. We can see that in the process of building the tree the examples were first split at the point a_1=6, and in the second stage the larger cluster of examples was again split at point a_1=3. If we write this tree in the form of rules, we get three rules that do not overlap and cover the entire example space (each rule covers three examples). We can also see that these rules predict the target attribute values of the SYNTH-1 data perfectly. Now let us consider the ordered rules (Figure 5.4). We only have two 'ordinary' rules and a default rule, of which every one again covers three examples. We can see that the first rule is the same as the first 'tree' rule, but the second rule has one condition less than the corresponding 'tree' rule, and the default rule has no condition at all. However, if we take into account that these rules are ordered and must be interpreted in a sequence, we see that each rule covers exactly the same part of the example space. As a consequence, ordered rules also predict the target attribute values of the SYNTH-1 data perfectly. Finally, let us examine the unordered rules (Figure 5.5). The learned unordered rules are exactly the same as the ordered ones, but since they are interpreted as unordered they no longer perfectly predict the values of the target attribute. Namely, the three examples with the target attribute value 3 are covered by two rules, each predicting a different target value. The final prediction is a combination of the two separate predictions, and is, depending on the weighting scheme we use for combining rule predictions, somewhere between 2 and 3, instead of the accurate value which is 3. Therefore, this simple example suggests that unordered regression rules induced with a covering algorithm could perform worse than regression trees even on simple monotonic regression problems. However, we have not noticed this effect in our experiments, which suggests that this effect is not significant in real life problem domains.

5.3 Comparison of multiple target to single target prediction

One of the main motives that lead us to the development of predictive clustering rules (PCRs) was the possibility to use them for multiple target prediction. The main benefit of multiple target prediction is that a set of models (rule sets) each predicting one target attribute can be replaced by just one model that predicts all target attributes at once. Assuming that the target attributes are correlated, the complexity of the multiple target model should be much lower than the complexity of the corresponding set of models. In this section, we evaluate and compare PCRs used for multiple target prediction and PCRs used for single target prediction. For each data set, we have learned one multiple target PCR model and compared it to a set of single target PCR models. This set consists of the same number of models as is the number of target attributes in a given domain. The sizes of the single target PCR rule sets for each target attribute are summed and compared to the size of the multiple target PCR rule set. The overall significance of differences is estimated using the *Wilcoxon signed-rank* test (Wilcoxon, 1945), where each target attribute of each data set corresponds to one data point. Additionally, we visually present the performance of all four types of PCRs (single and multiple target prediction, ordered and unordered rules) using the *average ranks diagrams* (Demšar, 2006) and the Nemenyi test (Nemenyi, 1963) (see Section 5.2.1 for explanation of how to read these diagrams). We also present some examples of rules learned in all four settings of the PCR algorithm. The comparison is done separately for classification and for regression.

Classification. The significances of differences between PCRs used for single target and multiple target prediction are given in Table 5.24. The error rates and rule set sizes are presented in Table 5.25 for ordered PCRs and in Table 5.26 for unordered

Table 5.24: Comparison of *error rates* of *PCR* algorithms for *ordered and unordered rules* used for *single target* and *multiple target classification. Significances* (p-values) of differences in error rates and rule set sizes over all data sets and all target attributes. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller error rate or size. Significant differences are typeset in bold.

Compared algorithms			error p-value	size p-value
PCR ORDERED:	SINGLE	MULTIPLE	0.066 <	**<0.001 >**
PCR UNORDERED:	SINGLE	MULTIPLE	0.067 >	**<0.001 >**

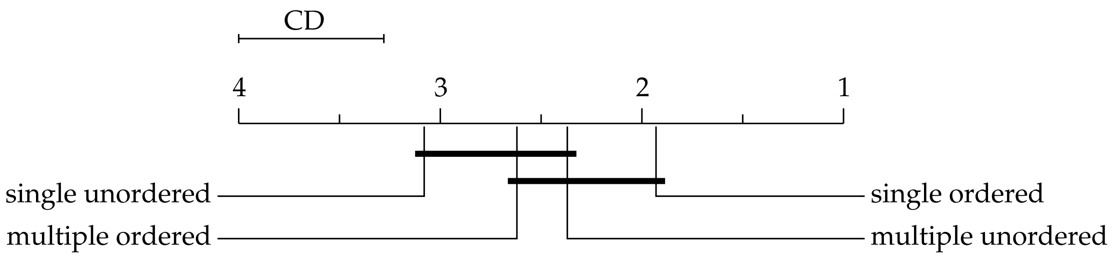


Figure 5.6: Average ranks diagram for *multiple target classification* problems, comparison of *error rates* of single and multiple target, ordered and unordered PCRs compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

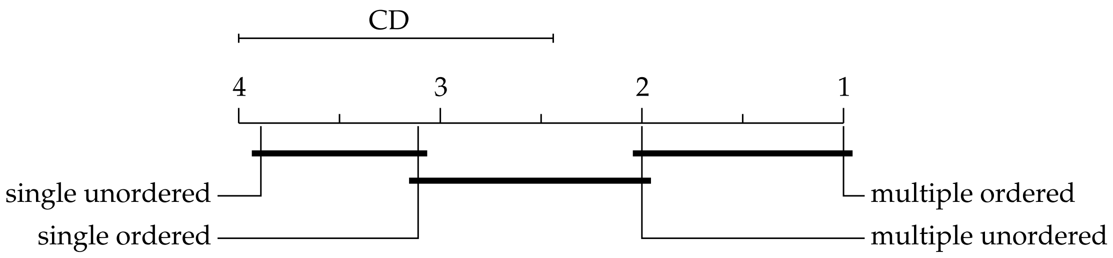


Figure 5.7: Average ranks diagram for *multiple target classification* problems, comparison of *rule set sizes* of single and multiple target, ordered and unordered PCRs compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

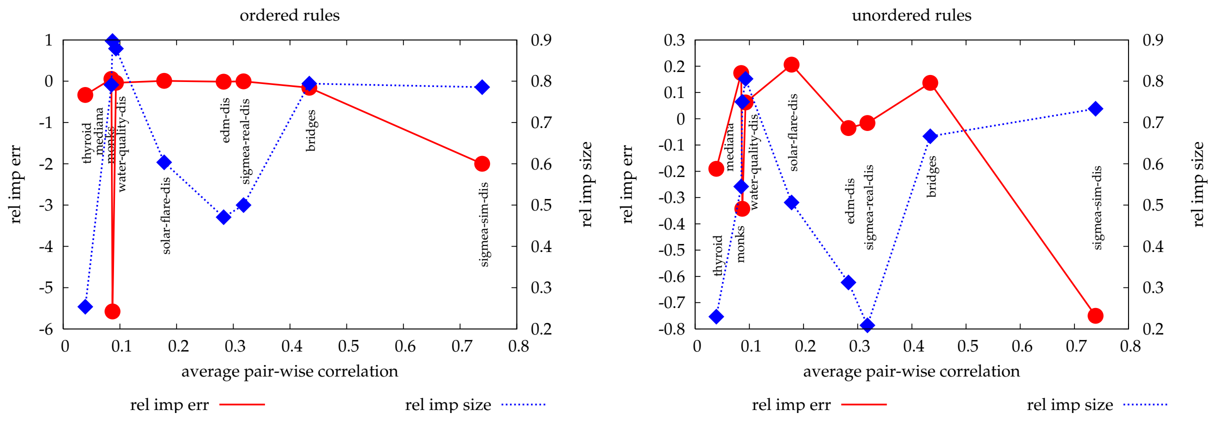


Figure 5.8: *Multiple target classification problems,* influence of the correlation between the target attributes on the relative improvement of error rates and rule set sizes.

Table 5.25: Comparison of *error rates* of *ordered PCRs* used for *single target* and *multiple target classification.* For each data set, the average error rate over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets are summed and compared to multiple target prediction sule set. In each row, the smallest error rate is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row (next page) gives the average error rate over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE			PCR MULTIPLE		
TAR. ATT.	% ERROR		# SIZE	% ERROR		# SIZE
BRIDGES	**35.0**		34	40.5		7
T-OR-D	**19.4**	±14.1	4	24.7	±0.0	
MATERIAL	21.6	±13.5	6	**20.0**	±10.1	
SPAN	**31.8**	±14.9	6	43.5	±11.3	
REL-L	47.5	±21.5	7	**44.7**	±0.0	
TYPE	**54.9**	±17.2	11	69.4	±0.0	
EDM-DIS	**24.7**		17	25.0		9
D-FLOW	13.6	±15.5	7	**11.7**	±8.1	
D-GAP	**35.7**	±11.8	10	38.3	±8.2	
MEDIANA	18.2		1297	**17.2**		271
READ-DELO	23.1	±0.9	306	**22.0**	±1.5	
READ-DNEVNIK	21.9	±1.2	436	**16.6**	±1.4	
READ-EKIPA	9.2	±0.9	296	**7.1**	±0.8	
READ-SL-NOV	**26.3**	±1.4	100	28.8	±1.1	
READ-VECER	**10.3**	±1.7	159	11.6	±0.9	
MONKS	**3.3**		39	21.7		4
MONK-1	**0.0**	±0.0	7	30.1	±9.1	
MONK-2	**10.0**	±6.4	28	33.1	±8.0	
MONK-3	**0.0**	±0.0	4	1.9	±3.0	
SIGMEA-REAL-DIS	**24.8**		76	24.9		38
MFO	26.1	±5.4	52	**24.5**	±5.2	
MSO	**23.5**	±6.1	24	25.3	±3.8	
SIGMEA-SIM-DIS	**0.7**		14	2.1		3
DISP-RATE	**1.4**	±0.4	12	4.3	±0.7	
DISP-SEEDS	**0.0**	±0.0	2	**0.0**	±0.0	
SOLAR-FLARE-DIS	11.1		58	**11.0**		23
C-CLASS	15.8	±7.6	25	**15.2**	±6.7	
M-CLASS	**13.0**	±3.7	19	14.6	±4.7	
X-CLASS	4.6	±4.1	14	**3.1**	±3.2	

Continued on the next page.

Table 5.25: Continued from the previous page.

DATA SET	PCR SINGLE			PCR MULTIPLE		
TAR. ATT.	% ERROR		# SIZE	% ERROR		# SIZE
THYROID-0387	**1.8**		666	2.4		497
HYPER-THYRO	**2.0**	±0.5	107	2.5	±0.6	
HYPO-THYRO	**0.9**	±0.4	53	3.1	±0.8	
BIND-PROT	**2.9**	±0.8	135	3.2	±0.8	
GEN-HEALTH	**2.6**	±0.8	125	3.6	±0.6	
REPL-THEORY	**2.1**	±0.5	129	**2.1**	±0.4	
ANTITHYRO-TR	**0.3**	±0.2	24	**0.3**	±0.2	
DISC-RESULTS	**1.6**	±0.3	93	2.0	±0.5	
WATER-QUALITY-DIS	**32.1**		736	33.3		89
CLAD-SP	**38.5**	±3.7	31	39.5	±5.6	
GONG-INC	34.4	±4.2	64	**29.5**	±3.6	
OEDO-SP	29.8	±3.6	62	**29.7**	±4.5	
TIGE-TEN	25.1	±2.6	80	**23.2**	±4.6	
MELO-VAR	**34.2**	±4.0	30	41.7	±4.8	
NITZ-PAL	**29.6**	±3.0	23	31.3	±2.3	
AUDO-CHA	30.5	±5.0	88	**29.3**	±5.0	
ERPO-OCT	31.7	±3.7	75	**29.0**	±2.6	
GAMM-FOSS	**32.3**	±3.8	27	38.1	±4.5	
BAET-RHOD	32.0	±4.8	49	**31.8**	±3.1	
HYDRO-SP	**34.9**	±4.5	29	39.1	±4.6	
RHYA-SP	**29.1**	±4.4	59	36.1	±5.5	
SIMU-SP	**37.6**	±4.5	59	38.5	±5.5	
TUBI-SP	29.2	±3.8	60	**28.8**	±4.0	
AVERAGE	**20.3**		326.3	22.6		104.6

PCRs. Additionally, Figures 5.6 and 5.7 present the average ranks of ordered and unordered PCRs for single and multiple target prediction as compared by error rates and rule set size. From Table 5.24 we can conclude that ordered PCRs are better in single target setting than in the multiple target setting (*p-value*=0.066). Unordered rules, on the other hand, are better in multiple target setting than in single target setting (*p-value*=0.067). We believe the reason for improvement of unordered rules in multiple target setting is that learning rules for several targets simultaneously enables a more cautious rule specialization and prevents unordered rules from overfitting. Namely, each rule must cover examples with homogeneous values of all target attributes, and not just one. This automatically decreases the problem of noise in the learning data, and also promotes more general rules. This effect should in principle be beneficial also for ordered rules. However, it seems that the single target ordered PCRs are very well

Table 5.26: Comparison of *error rates* of *unordered PCRs* used for *single target* and *multiple target classification*. For each data set, the average error rate over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets and summed and compared to multiple target prediction sule set. In each row, the smallest error rate is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row (next page) gives the average error rate over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE			PCR MULTIPLE		
TAR. ATT.	% ERROR		# SIZE	% ERROR		# SIZE
BRIDGES	37.3		36	**32.2**		12
T-OR-D	19.4	±14.1	4	**10.6**	±8.7	
MATERIAL	26.5	±17.8	6	**18.8**	±10.4	
SPAN	**35.2**	±13.3	6	40.0	±11.0	
REL-L	46.5	±20.6	8	**35.3**	±15.7	
TYPE	58.8	±12.7	12	**56.5**	±0.0	
EDM-DIS	**28.2**		16	29.2		11
D-FLOW	16.2	±15.2	7	**12.3**	±9.0	
D-GAP	**40.3**	±0.0	9	46.1	±13.4	
MEDIANA	20.1		1505	**16.6**		685
READ-DELO	24.0	±1.3	353	**21.7**	±1.2	
READ-DNEVNIK	20.4	±1.7	493	**15.4**	±0.9	
READ-EKIPA	7.4	±0.9	362	**6.3**	±0.8	
READ-SL-NOV	38.0	±4.6	100	**29.2**	±0.9	
READ-VECER	10.5	±1.1	197	**10.4**	±1.0	
MONKS	**17.5**		40	23.5		10
MONK-1	**11.1**	±4.9	7	17.6	±7.3	
MONK-2	**33.3**	±13.1	29	35.9	±5.9	
MONK-3	**8.1**	±7.2	4	17.1	±5.4	
SIGMEA-REAL-DIS	**24.5**		91	24.9		72
MFO	26.2	±5.8	60	**25.1**	±4.6	
MSO	**22.8**	±5.3	31	24.6	±3.3	
SIGMEA-SIM-DIS	**1.2**		15	2.1		4
DISP-RATE	**2.4**	±0.7	13	4.3	±0.7	
DISP-SEEDS	**0.0**	±0.0	2	**0.0**	±0.0	
SOLAR-FLARE-DIS	13.1		79	**10.4**		39
C-CLASS	18.3	±8.2	36	**13.6**	±6.9	
M-CLASS	15.8	±4.0	27	**14.9**	±4.6	
X-CLASS	5.3	±5.3	16	**2.8**	±3.4	

Continued on the next page.

Table 5.26: Continued from the previous page.

DATA SET TAR. ATT.	PCR SINGLE % ERROR		# SIZE	PCR MULTIPLE % ERROR		# SIZE
THYROID-0387	**2.1**		727	2.5		560
HYPER-THYRO	**1.7**	±0.5	115	2.5	±0.5	
HYPO-THYRO	**2.4**	±0.7	40	3.7	±0.5	
BIND-PROT	**3.0**	±0.6	157	3.4	±0.6	
GEN-HEALTH	3.0	±0.7	134	**2.7**	±0.9	
REPL-THEORY	**2.2**	±0.7	148	3.0	±0.8	
ANTITHYRO-TR	**0.4**	±0.2	24	**0.4**	±0.2	
DISC-RESULTS	2.1	±0.6	109	**2.0**	±0.6	
WATER-QUALITY-DIS	33.9		788	**31.8**		153
CLAD-SP	**39.9**	±4.9	28	40.4	±4.9	
GONG-INC	39.2	±6.6	77	**28.4**	±3.2	
OEDO-SP	30.9	±4.7	78	**29.9**	±5.1	
TIGE-TEN	23.9	±4.8	83	**20.8**	±2.4	
MELO-VAR	**38.9**	±3.3	27	41.4	±3.7	
NITZ-PAL	**30.0**	±4.8	25	30.6	±4.3	
AUDO-CHA	30.8	±5.9	90	**24.2**	±5.2	
ERPO-OCT	32.7	±4.3	79	**26.5**	±3.3	
GAMM-FOSS	**33.0**	±5.9	23	37.8	±3.7	
BAET-RHOD	33.5	±5.4	57	**32.5**	±2.4	
HYDRO-SP	39.1	±4.2	33	**38.2**	±4.9	
RHYA-SP	32.8	±4.2	69	**30.8**	±6.8	
SIMU-SP	39.3	±4.6	55	**37.3**	±4.7	
TUBI-SP	31.0	±3.9	64	**27.1**	±3.1	
AVERAGE	22.7		366.3	**21.4**		171.8

suited for the problems used in our (single vs. multiple target) experimental evaluation. Namely, from the diagram in Figure 5.6 we can see that on these data sets, single target ordered PCRs significantly outperform single target unordered PCRs, while in the comparison in Section 5.2.1 unordered PCRs were significantly better than ordered PCRs. Therefore, we believe the more cautious specialization introduced by the multiple target learning prevented ordered rules from achieving maximal accuracy.

Now if we consider the rule set sizes (Table 5.24 and Figure 5.7), we see that multiple target PCR rule sets are significantly smaller the the corresponding single target rule sets (*p-value*<0.001). This finding confirms our hypothesis that the PCR algorithm can indeed learn multiple target classification rule sets that are significantly smaller than the corresponding single target rule sets. From Figure 5.7 we can also see that ordered multiple target rules are always smaller (average rank is 1) than

Table 5.27: An example of *ordered* single target and multiple target PCR *classification* rules learned on the SIGMEA-SIM-DIS domain.

Ordered single target rules : *disp_rate*

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_rate = 1]$

Rule 2: IF $(distance > 100) \wedge (area\_d \leq 10,000)$ THEN $[disp\_rate = 0]$

Rule 3: IF $(distance \leq 50) \wedge (ratio\_l\_wr > 0)$ THEN $[disp\_rate = 1]$

Rule 4: IF $(circumfer\_d \leq 20) \wedge (distance > 10)$ THEN $[disp\_rate = 0]$

Rule 5: IF $(ratio\_l\_wd > 0.001) \wedge (distance \leq 500) \wedge (circum\_r \leq 1,924)$ THEN $[disp\_rate = 1]$

Rule 6: IF $(distance > 50)$ THEN $[disp\_rate = 0]$

Rule 7: IF $(distance \leq 10) \wedge (donating\_plot > 955) \wedge (circumfer\_d > 13.9)$ THEN $[disp\_rate = 1]$

Rule 8: IF $(area\_d > 100) \wedge (ratio\_l\_wd > 0.001) \wedge (donating\_plot > 2,731)$ THEN $[disp\_rate = 1]$

Rule 9: IF $(orientation = 1) \wedge (distance > 0)$ THEN $[disp\_rate = 0]$

Rule 10: IF $(distance \leq 0)$ THEN $[disp\_rate = 1]$

Rule 11: IF $(area\_d > 100)$ THEN $[disp\_rate = 1]$

Rule 12: IF $(donating\_plot > 953)$ THEN $[disp\_rate = 0]$

Otherwise: $[disp\_rate = 1]$

Error rate = 1.4%

Ordered single target rules : *disp_seeds*

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_seeds = 1]$

Rule 2: IF $(distance > 0)$ THEN $[disp\_seeds = 0]$

Otherwise: $[disp\_seeds = 1]$

Error rate = 0%

Average error rate = 0.7%

Ordered multiple target rules

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_rate = 1, disp\_seeds = 1]$

Rule 2: IF $(distance > 50)$ THEN $[disp\_rate = 0, disp\_seeds = 0]$

Rule 3: IF $(distance > 0)$ THEN $[disp\_rate = 1, disp\_seeds = 0]$

Otherwise: $[disp\_rate = 1, disp\_seeds = 1]$

Error rate = $[disp\_rate = 4.3\%, disp\_seeds = 0\%]$

Average error rate = 2.1%

Table 5.28: An example of *unordered* single target and multiple target PCR *classification* rules learned on the SIGMEA-SIM-DIS domain.

Unordered single target rules : *disp_rate*

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_rate = 1]$

Rule 2: IF $(distance > 100) \land (area\_d \leq 10,000)$ THEN $[disp\_rate = 0]$

Rule 3: IF $(ratio\_l\_wr > 0) \land (distance \leq 50)$ THEN $[disp\_rate = 1]$

Rule 4: IF $(circumfer\_d \leq 20) \land (distance > 10)$ THEN $[disp\_rate = 0]$

Rule 5: IF $(distance \leq 500) \land (ratio\_l\_wd > 0.001) \land (circum\_r \leq 1,924)$ THEN $[disp\_rate = 1]$

Rule 6: IF $(distance > 50) \land (donating\_plot \leq 4,940)$ THEN $[disp\_rate = 0]$

Rule 7: IF $(distance \leq 10) \land (donating\_plot > 955) \land (circumfer\_d > 13.9)$ THEN $[disp\_rate = 1]$

Rule 8: IF $(area\_d > 100) \land (circumfer\_d \leq 461.9) \land (donating\_plot > 2,731)$ THEN $[disp\_rate = 1]$

Rule 9: IF $(orientation = 1) \land (distance > 0)$ THEN $[disp\_rate = 0]$

Rule 10: IF $(ratio\_l\_wr \leq 0) \land (area\_d > 100)$ THEN $[disp\_rate = 1]$

Rule 11: IF $(distance > 10)$ THEN $[disp\_rate = 0]$

Rule 12: IF $(distance \leq 0)$ THEN $[disp\_rate = 1]$

Rule 13: IF $(donating\_plot > 89)$ THEN $[disp\_rate = 1]$

Otherwise: $[disp\_rate = 0]$

Error rate = 2.4%

Unordered single target rules : *disp_seeds*

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_seeds = 1]$

Rule 2: IF $(distance > 0)$ THEN $[disp\_seeds = 0]$

Otherwise: $[disp\_seeds = 1]$

Error rate = 0%

Average error rate = 1.2%

Unordered multiple target rules

Rule 1: IF $(donor\_equals\_reciever = true)$ THEN $[disp\_rate = 1, disp\_seeds = 1]$

Rule 2: IF $(distance > 50)$ THEN $[disp\_rate = 0, disp\_seeds = 0]$

Rule 3: IF $(distance > 0)$ THEN $[disp\_rate = 1, disp\_seeds = 0]$

Rule 4: IF $(distance \leq 0)$ THEN $[disp\_rate = 1, disp\_seeds = 1]$

Otherwise: $[disp\_rate = 0, disp\_seeds = 0]$

Error rate = $[disp\_rate = 4.3\%, disp\_seeds = 0\%]$

Average error rate = 2.1%

---

unordered multiple target rules (with average rank 2). This speaks in favor of our previously stated hypothesis, that the ordered rules accuracy is worse in the multiple target than in the single target setting because ordered multiple target rules are overly small.

One of our initial hypotheses was, that multiple target learning should be more appropriate and efficient on problem domains with target attributes that are more strongly correlated. In order to test this we have plotted the relative improvement[5] in average error rates and rule sets sizes against the average pair-wise correlation[6] between the target attributes in each domain. These plots separately for ordered and unordered rules are given in Figure 5.8. If our hypothesis is correct, the relative improvement in error rates and rule set sizes (when switching from single target to multiple target prediction) should increase with increasing average pair-wise correlation between the target attributes in each domain. However, even with a lot of imagination, it is hard to observe such a trend. The absence of such a trend suggests that the correlation between target attributes in the entire example space is not a key factor in the effectiveness of multiple target PCR learning. We believe that multiple target PCRs perform well on domains where clusters of examples can be found such that within each cluster the target attributes are correlated. It is therefore not necessary for the target attributes to be correlated within the entire example space. We can illustrate this by inspecting an example of single and multiple target PCRs; ordered rules learned on the SIGMEA-SIM-DIS are given in Figure 5.27 and unordered in Figure 5.28. If we compare ordered single and multiple target rule sets (or unordered, since they are both very similar) we can see that the first rule in both single target rule sets, and in the multiple target rule set is the same. This means that both target attributes demand a similar clustering of the attribute space in order to describe them well. A counter example is the MONKS domain (Section 5.1.3). Its target attributes assume completely different clustering of the attribute space, and as a consequence, multiple target PCRs are much less accurate than single target PCRs (Tables 5.25 and 5.26).

Regression. The significances of differences between the PCRs used for single target and multiple target prediction are given in Table 5.29. The relative root mean squared errors (RRMSE), correlation coefficients, and rule set sizes are presented in Tables 5.30 and 5.31 for ordered PCRs, and in Tables 5.32 and 5.33 for unordered PCRs. In addition, Figures 5.9, 5.10 and 5.11 present the average ranks of ordered and

[5]The relative improvement in error rates, for example, was calculated as $1 - \overline{\text{err}}_{multiple}/\overline{\text{err}}_{single}$.

[6]The correlation between two nominal attributes was measured with the Cramér's V coefficient.

Table 5.29: Comparison of *RRMSE* and *correlation coefficients* of *PCR* algorithms for *ordered and unordered rules* used for *single target* and *multiple target regression. Significances* (p-values) of differences in RRMSE, correlation coefficients, and rule set sizes over all data sets and all target attributes. The sign < (>) right of a p-value means that the first (second) algorithm tends to induce rule sets with smaller RRMSE, correlation coefficients, or rule set sizes. Significant differences are typeset in bold.

Compared algorithms			RRMSE P-VALUE	CORR P-VALUE	SIZE P-VALUE
PCR ORDERED:	SINGLE	MULTIPLE	**<0.001 >**	0.970 <	**0.002 >**
PCR UNORDERED:	SINGLE	MULTIPLE	0.461 >	**0.042 >**	**<0.001 >**

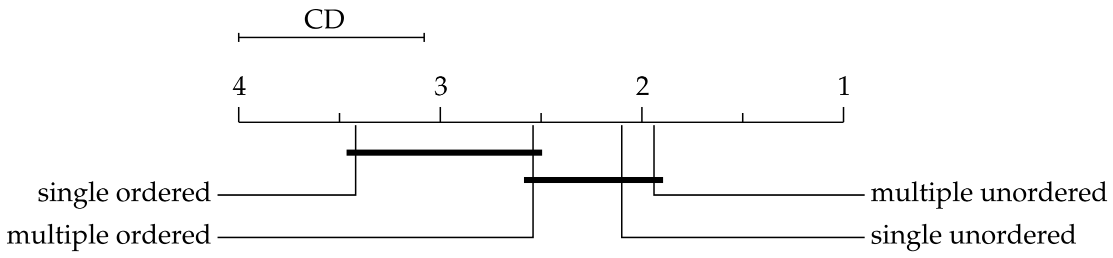


Figure 5.9: Average ranks diagram for *multiple target regression* problems, comparison of *RRMSE* of single/multiple target, ordered/unordered PCRs compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

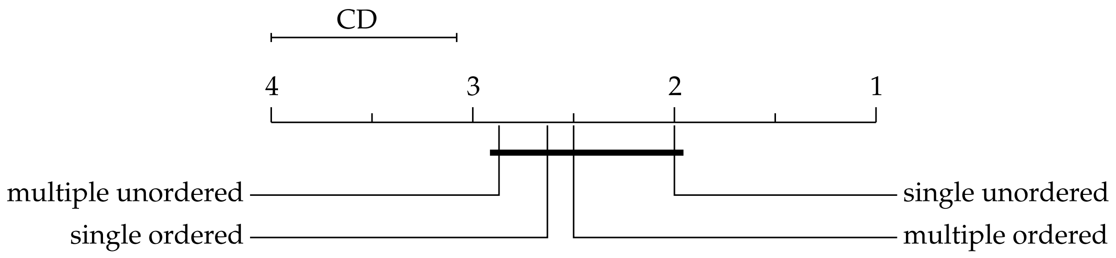


Figure 5.10: Average ranks diagram for *multiple target regression* problems, comparison of *correlation coefficients* of single/multiple target, ordered/unordered PCRs compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

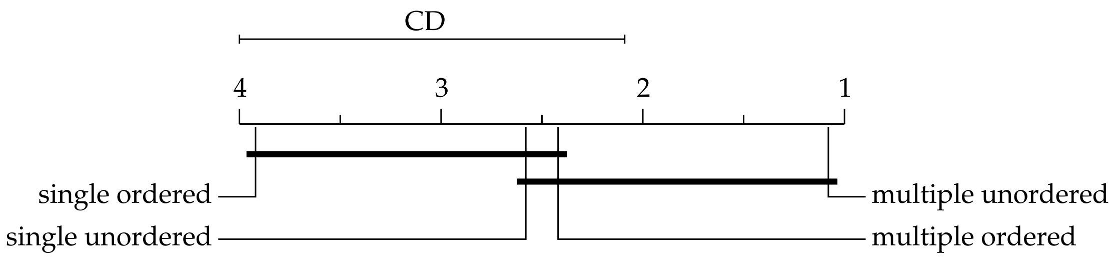


Figure 5.11: Average ranks diagram for *multiple target regression* problems, comparison of *rule set sizes* of single/multiple target, ordered/unordered PCRs compared to each other with the Nemenyi test. Algorithms that do not differ significantly (*p-value*=0.05) are connected.

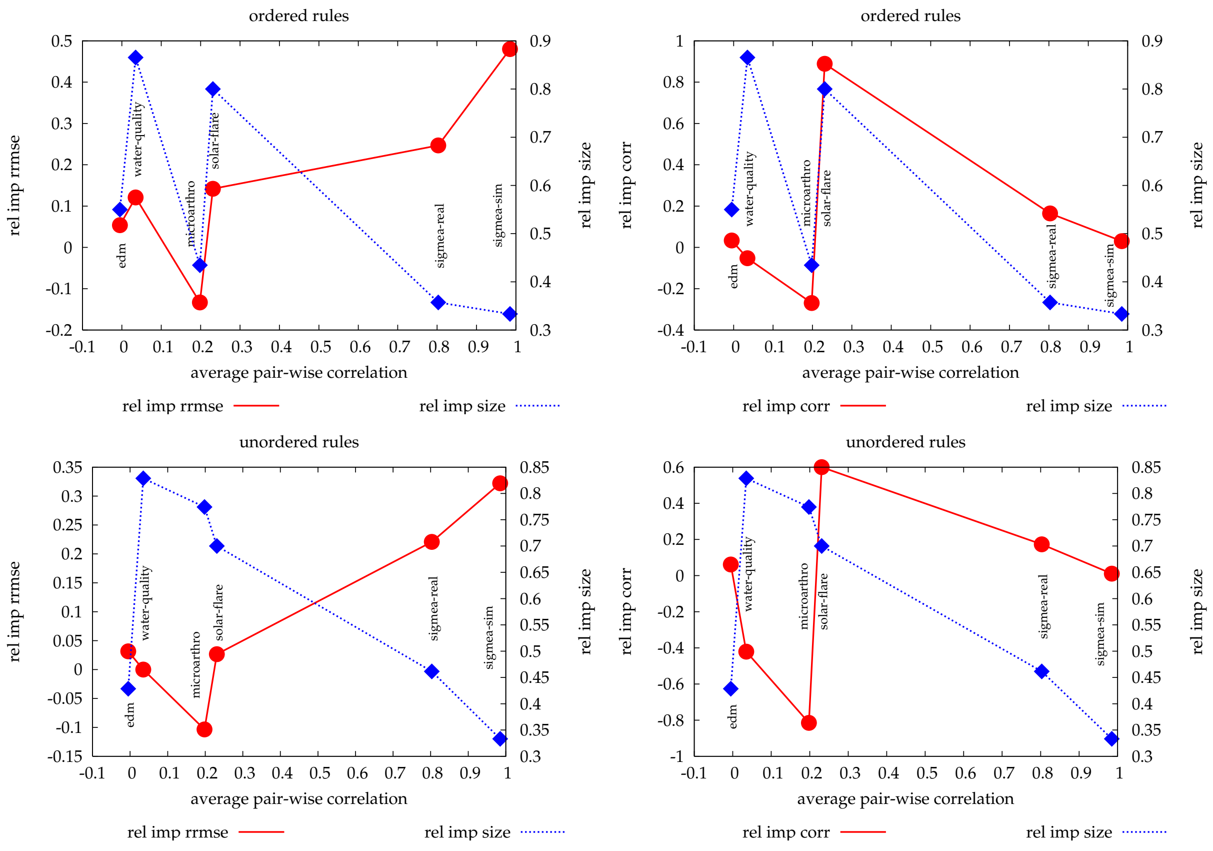


Figure 5.12: *Multiple target regression problems,* influence of target attribute correlation on the relative improvement in RRMSE, correlation coefficients, and rule set sizes.

Table 5.30: Comparison of *RRMSE* of *ordered PCRs* used for *single target* and *multiple target regression.* For each data set, the average RRMSE over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets and summed and compared to multiple target prediction sule set. In each row, the smallest RRMSE is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average RRMSE over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE		PCR MULTIPLE	
TAR. ATT.	RRMSE	# SIZE	RRMSE	# SIZE
EDM	0.93	20	**0.88**	9
D-FLOW	0.97 ±0.47	9	**0.94** ±0.43	
D-GAP	0.89 ±0.10	11	**0.82** ±0.09	
MICROARTHROPODS	**0.75**	191	0.85	108
ACARI	**0.75** ±0.15	35	0.89 ±0.17	
COLLEMBOLAN	**0.75** ±0.13	48	0.90 ±0.19	
SH-BIODIV	**0.76** ±0.04	108	**0.76** ±0.04	
SIGMEA-REAL	0.81	14	**0.61**	9
MFO	1.06 ±0.65	8	**0.74** ±0.47	
MSO	0.56 ±0.39	6	**0.49** ±0.32	
SIGMEA-SIM	0.25	3	**0.13**	2
DISP-RATE	**0.15** ±0.00	2	**0.15** ±0.00	
DISP-SEEDS	0.35 ±0.01	1	**0.11** ±0.00	
SOLAR-FLARE	1.34	65	**1.15**	13
C-CLASS	1.32 ±0.31	28	**1.04** ±0.30	
M-CLASS	1.39 ±0.53	25	**1.09** ±0.36	
X-CLASS	**1.32** ±0.81	12	**1.32** ±0.77	
WATER-QUALITY	1.24	1375	**1.09**	185
CLAD-SP	1.27 ±0.11	124	**1.10** ±0.10	
GONG-INC	1.42 ±0.09	122	**1.14** ±0.11	
OEDO-SP	1.37 ±0.18	95	**1.17** ±0.18	
TIGE-TEN	1.21 ±0.22	79	**1.08** ±0.10	
MELO-VAR	1.24 ±0.00	86	**1.09** ±0.05	
NITZ-PAL	1.13 ±0.10	89	**0.98** ±0.08	
AUDO-CHA	1.26 ±0.00	72	**1.19** ±0.00	
ERPO-OCT	1.27 ±0.18	93	**1.06** ±0.09	
GAMM-FOSS	1.09 ±0.06	87	**1.00** ±0.07	
BAET-RHOD	1.21 ±0.17	112	**1.05** ±0.00	
HYDRO-SP	1.23 ±0.11	114	**1.11** ±0.08	
RHYA-SP	1.25 ±0.16	103	**1.12** ±0.12	
SIMU-SP	1.28 ±0.16	108	**1.15** ±0.00	
TUBI-SP	1.14 ±0.11	91	**1.00** ±0.11	
AVERAGE	1.06	278.0	**0.94**	54.3

Table 5.31: Comparison of *correlation cofficients* of *ordered PCRs* used for *single target* and *multiple target regression.* For each data set, the average correlation coefficient over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets and summed and compared to multiple target prediction sule set. In each row, the largest correlation coefficient is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average correlation coefficient over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE		PCR MULTIPLE	
TAR. ATT.	CORR	# SIZE	CORR	# SIZE
EDM	0.58	20	**0.60**	9
D-FLOW	0.56 ±0.47	9	**0.61** ±0.39	
D-GAP	**0.60** ±0.14	11	0.59 ±0.13	
MICROARTHROPODS	**0.66**	191	0.52	108
ACARI	**0.66** ±0.11	35	0.47 ±0.09	
COLLEMBOLAN	**0.66** ±0.05	48	0.43 ±0.08	
SH-BIODIV	**0.66** ±0.06	108	**0.66** ±0.06	
SIGMEA-REAL	0.66	14	**0.79**	9
MFO	0.48 ±0.41	8	**0.72** ±0.24	
MSO	0.83 ±0.33	6	**0.87** ±0.26	
SIGMEA-SIM	0.96	3	**0.99**	2
DISP-RATE	**0.99** ±0.00	2	**0.99** ±0.00	
DISP-SEEDS	0.94 ±0.00	1	**0.99** ±0.00	
SOLAR-FLARE	0.01	65	**0.09**	13
C-CLASS	0.03 ±0.16	28	**0.17** ±0.31	
M-CLASS	0.03 ±0.24	25	**0.12** ±0.19	
X-CLASS	-0.03 ±0.03	12	**-0.03** ±0.03	
WATER-QUALITY	**0.20**	1375	0.19	185
CLAD-SP	**0.20** ±0.08	124	0.13 ±0.09	
GONG-INC	**0.03** ±0.05	122	-0.02 ±0.09	
OEDO-SP	0.07 ±0.08	95	**0.12** ±0.11	
TIGE-TEN	0.23 ±0.16	79	**0.27** ±0.12	
MELO-VAR	0.18 ±0.00	86	**0.19** ±0.12	
NITZ-PAL	0.31 ±0.11	89	**0.34** ±0.11	
AUDO-CHA	**0.16** ±0.00	72	0.11 ±0.00	
ERPO-OCT	0.18 ±0.15	93	**0.24** ±0.12	
GAMM-FOSS	**0.37** ±0.06	87	0.31 ±0.10	
BAET-RHOD	**0.24** ±0.14	112	0.21 ±0.00	
HYDRO-SP	0.18 ±0.11	114	0.12 ±0.09	
RHYA-SP	0.19 ±0.12	103	**0.20** ±0.07	
SIMU-SP	**0.15** ±0.16	108	0.07 ±0.00	
TUBI-SP	0.31 ±0.10	91	**0.33** ±0.15	
AVERAGE	**0.35**	278.0	**0.35**	54.3

Table 5.32: Comparison of *RRMSE* of *unordered PCRs* used for *single target* and *multiple target regression.* For each data set, the average RRMSE over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets and summed and compared to multiple target prediction sule set. In each row, the smallest RRMSE is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average RRMSE over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE		PCR MULTIPLE	
TAR. ATT.	RRMSE	# SIZE	RRMSE	# SIZE
EDM	0.95	7	**0.92**	4
D-FLOW	**1.03** ±0.43	5	1.04 ±0.37	
D-GAP	0.86 ±0.13	2	**0.80** ±0.09	
MICROARTHROPODS	**0.87**	31	0.96	7
ACARI	**0.84** ±0.18	11	0.99 ±0.22	
COLLEMBOLAN	**0.85** ±0.19	13	0.96 ±0.20	
SH-BIODIV	**0.93** ±0.06	7	**0.93** ±0.06	
SIGMEA-REAL	0.77	13	**0.60**	7
MFO	0.98 ±0.49	7	**0.70** ±0.38	
MSO	0.55 ±0.35	6	**0.50** ±0.30	
SIGMEA-SIM	0.59	3	**0.40**	2
DISP-RATE	**0.44** ±0.01	2	**0.44** ±0.00	
DISP-SEEDS	0.75 ±0.02	1	**0.35** ±0.00	
SOLAR-FLARE	1.13	40	**1.10**	12
C-CLASS	1.11 ±0.27	13	**1.02** ±0.31	
M-CLASS	1.08 ±0.37	19	**1.05** ±0.36	
X-CLASS	**1.20** ±0.78	8	1.24 ±0.74	
WATER-QUALITY	**0.99**	181	**0.99**	31
CLAD-SP	**0.99** ±0.06	14	**0.99** ±0.00	
GONG-INC	1.04 ±0.09	12	**1.01** ±0.10	
OEDO-SP	**0.99** ±0.11	11	**0.99** ±0.10	
TIGE-TEN	0.98 ±0.00	9	**0.97** ±0.16	
MELO-VAR	**1.00** ±0.08	15	**1.00** ±0.08	
NITZ-PAL	**0.93** ±0.07	12	0.96 ±0.06	
AUDO-CHA	1.02 ±0.13	13	**1.00** ±0.13	
ERPO-OCT	1.00 ±0.11	14	**0.98** ±0.13	
GAMM-FOSS	**0.95** ±0.06	8	0.96 ±0.05	
BAET-RHOD	1.00 ±0.11	21	**0.99** ±0.13	
HYDRO-SP	**0.99** ±0.07	12	1.00 ±0.07	
RHYA-SP	**0.97** ±0.15	6	0.99 ±0.14	
SIMU-SP	**1.01** ±0.06	17	1.03 ±0.05	
TUBI-SP	**0.94** ±0.11	17	**0.94** ±0.12	
AVERAGE	0.94	45.8	**0.92**	10.5

Table 5.33: Comparison of *correlation cofficients* of *unordered PCRs* used for *single target* and *multiple target regression*. For each data set, the average correlation coefficient over all target attributes is given first, and then for each target attribute separately. Sizes of single target prediction rule sets and summed and compared to multiple target prediction sule set. In each row, the largest correlation coefficient is typeset in bold. Size is given as the number of learned rules, trees are converted to rules. The final row gives the average correlation coefficient over all target attributes of all data sets and the average rule set size over all data sets.

DATA SET	PCR SINGLE		PCR MULTIPLE	
TAR. ATT.	CORR	# SIZE	CORR	# SIZE
EDM	0.46	7	**0.49**	4
D-FLOW	**0.39** ±0.41	5	**0.39** ±0.32	
D-GAP	0.52 ±0.22	2	**0.60** ±0.13	
MICROARTHROPODS	**0.49**	31	0.27	7
ACARI	**0.55** ±0.06	11	0.17 ±0.08	
COLLEMBOLAN	**0.54** ±0.07	13	0.28 ±0.08	
SH-BIODIV	**0.37** ±0.07	7	**0.37** ±0.07	
SIGMEA-REAL	0.67	13	**0.81**	7
MFO	0.50 ±0.37	7	**0.75** ±0.24	
MSO	0.84 ±0.30	6	**0.87** ±0.25	
SIGMEA-SIM	0.94	3	**0.95**	2
DISP-RATE	**0.94** ±0.00	2	**0.94** ±0.00	
DISP-SEEDS	0.94 ±0.00	1	**0.97** ±0.00	
SOLAR-FLARE	0.04	40	**0.10**	12
C-CLASS	0.06 ±0.22	13	**0.19** ±0.30	
M-CLASS	0.09 ±0.18	19	**0.15** ±0.16	
X-CLASS	**-0.03** ±0.03	8	-0.04 ±0.04	
WATER-QUALITY	**0.27**	181	0.19	31
CLAD-SP	**0.27** ±0.08	14	0.19 ±0.00	
GONG-INC	**0.10** ±0.06	12	0.04 ±0.08	
OEDO-SP	**0.23** ±0.06	11	0.20 ±0.10	
TIGE-TEN	**0.28** ±0.00	9	0.26 ±0.16	
MELO-VAR	**0.24** ±0.12	15	0.13 ±0.07	
NITZ-PAL	**0.41** ±0.08	12	0.30 ±0.12	
AUDO-CHA	**0.16** ±0.14	13	0.12 ±0.08	
ERPO-OCT	**0.25** ±0.07	14	0.23 ±0.14	
GAMM-FOSS	**0.37** ±0.10	8	0.30 ±0.08	
BAET-RHOD	**0.27** ±0.12	21	0.18 ±0.10	
HYDRO-SP	**0.26** ±0.05	12	0.13 ±0.09	
RHYA-SP	**0.30** ±0.08	6	0.21 ±0.14	
SIMU-SP	**0.20** ±0.14	17	0.06 ±0.10	
TUBI-SP	**0.38** ±0.09	17	0.36 ±0.13	
AVERAGE	**0.36**	45.8	0.32	10.5

Table 5.34: An example of *ordered* single target and multiple target PCR *regression* rules learned on the SIGMEA-SIM domain.

Ordered single target rules : *disp_rate*

Rule 1: IF $(donor\_equals\_reciever = false)$ THEN $[disp\_rate = 0.0057]$

Rule 2: IF $(ratio\_l\_wd > 0.11) \wedge (circumfer\_d > 13.9)$ THEN $[disp\_rate = 0.88]$

Otherwise: $[disp\_rate = 0.40]$

RRMSE = 0.15, Corr. coeff. = 0.99

Ordered single target rules : *disp_seeds*

Rule 1: IF $(donor\_equals\_reciever = false)$ THEN $[disp\_rate = 0.0042]$

Otherwise: $[disp\_seeds = 0.75]$

RRMSE = 0.35, Corr. coeff. = 0.94

Average RRMSE = 0.25, Average corr. coeff. = 0.96

Ordered multiple target rules

Rule 1: IF $(donor\_equals\_reciever = false)$ THEN $[disp\_rate = 0.0057, disp\_seeds = 0.0042]$

Rule 2: IF $(ratio\_l\_wd > 0.11) \wedge (area\_d > 9)$ THEN $[disp\_rate = 0.88, disp\_seeds = 0.93]$

Otherwise: $[disp\_rate = 0.39, disp\_seeds = 0.56]$

RRMSE = $[disp\_rate = 0.15, disp\_seeds = 0.11]$, Corr. coeff. = $[disp\_rate = 0.99, disp\_seeds = 0.99]$

Average RRMSE = 0.13, Average corr. coeff. = 0.99

unordered PCRs for single and multiple target prediction as compared by RRMSE, correlation coefficient, and rule set size. From Table 5.29, it follows that ordered multiple target PCRs have a significantly lower RRMSE than the single target ones (*p-value*<0.001). However, if we look at correlation coefficients, we see virtually no difference (*p-value*=0.97). In the case of unordered rules, the RRMSE and correlation coefficient show contradictory picture; judging by the correlation coefficient single target PCRs are better (*p-value*=0.042), while the RRMSE shows a very small difference in the opposite direction (*p-value*=0.461). Based on these conflicting and rather confusing results from two different error measures, one can hardly make any solid claims about the accuracy of the four types of PCRs; perhaps the most sound conclusion is that there are no distinctive differences between the four.

When comparing rule set sizes of single and multiple target regression PCRs, we get a much clearer picture (Table 5.29 and Figure 5.11). Multiple target rule sets

Table 5.35: An example of *unordered* single target and multiple target PCR *regression* rules learned on the SIGMEA-SIM domain.

Unordered single target rules : *disp_rate*

Rule 1: `IF` $(donor\_equals\_reciever = false)$ `THEN` $[disp\_rate = 0.0057]$

Rule 2: `IF` $(ratio\_l\_wd > 0.11) \wedge (circumfer\_d > 13.9)$ `THEN` $[disp\_rate = 0.88]$

Otherwise: $[disp\_rate = 0.64]$

RRMSE = 0.44, Corr. coeff. = 0.94

Unordered single target rules : *disp_seeds*

Rule 1: `IF` $(donor\_equals\_reciever = false)$ `THEN` $[disp\_rate = 0.0042]$

Otherwise: $[disp\_seeds = 0.38]$

RRMSE = 0.75, Corr. coeff. = 0.94

Average RRMSE = 0.59, Average corr. coeff. = 0.94

Unordered multiple target rules

Rule 1: `IF` $(donor\_equals\_reciever = false)$ `THEN` $[disp\_rate = 0.0057, disp\_seeds = 0.0042]$

Rule 2: `IF` $(ratio\_l\_wd > 0.11) \wedge (area\_d > 9)$ `THEN` $[disp\_rate = 0.88, disp\_seeds = 0.93]$

Otherwise: $[disp\_rate = 0.64, disp\_seeds = 0.75]$

RRMSE = $[disp\_rate = 0.44, disp\_seeds = 0.35]$, Corr. coeff. = $[disp\_rate = 0.94, disp\_seeds = 0.97]$

Average RRMSE = 0.40, Average corr. coeff. = 0.95

are significantly smaller (*p-value*=0.002 for ordered and *p-value*<0.001 for unordered rules). The average ranks diagram also shows that unordered rule sets are always smaller than their single/multiple target counterparts, and that there is only a small difference in size between multiple target ordered and single target unordered rule sets. Assuming that the above stated conclusion that all four types of PCRs are of comparable accuracy, one should prefer multiple target unordered rules, since they are the smallest.

As in the case of classification, we can test the connection of the improvement that the multiple target prediction achieves in a domain, to the correlation[7] of its target attributes, also for regression. The separate plots for ordered and unordered rules are presented in Figure 5.12. The trend supporting our hypothesis can be seen

[7]The correlation between the numeric attributes was estimated with the Pearson's correlation coefficient (Equation 3.25).

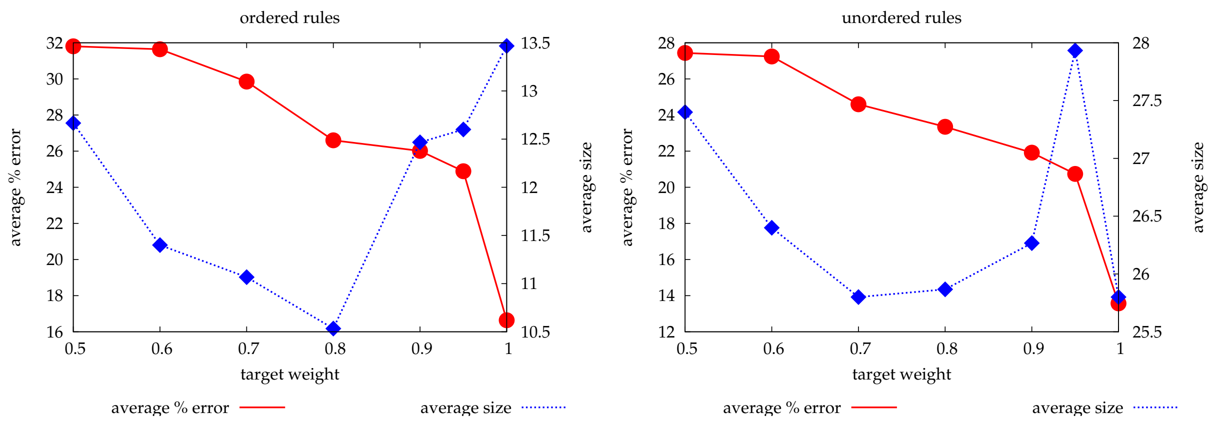


Figure 5.13: *Single target classification problems,* influence of target attribute weights on average error rate and rule set size for ordered and unordered PCRs. Averages are computed over all single target classification data sets.

for relative improvement of RRMSE for both ordered and unordered rules. For the relative improvements of correlation coefficients and rule sets size, however, the trends are more or less the opposite.

For the sake of completeness, let us present also an example of single and multiple target regression PCRs. Ordered and unordered rules learned on the original, non-discretized version of the SIGMEA-SIM data set are given in Figures 5.27 and 5.28. Again, ordered and unordered rule sets are very similar. The reduction of rule set size is smaller than in the case of classification, because the single target rule set modeling the *disp_rate* already comprises a small number of rules. As before, the first rule in single target rule sets and in the multiple target rule set are the same, meaning that both target attributes demand similar clustering of the attribute space. Interestingly, though, the attribute that appears in this rule is different from the one that appears in the first rules of the classification rule sets.

5.4 Influence of target attribute weights

The search heuristic that guides the single rule learning process (see Section 4.3) can take into account the target, as well as the non-target attributes, i.e., the descriptive attributes. The proportion of influence of each group of attributes can be set by means of the *target weight parameter* τ (see Equation 4.3). It should be set to a value from the $(0, 1]$ interval. In principle, its influence should be as follows. If our goal is to learn rules that are as accurate on the learning set as possible then we should set the parameter to one. On the other hand, setting the parameter to less than one should lead to rules that cover examples that are more similar with regard to the values of all

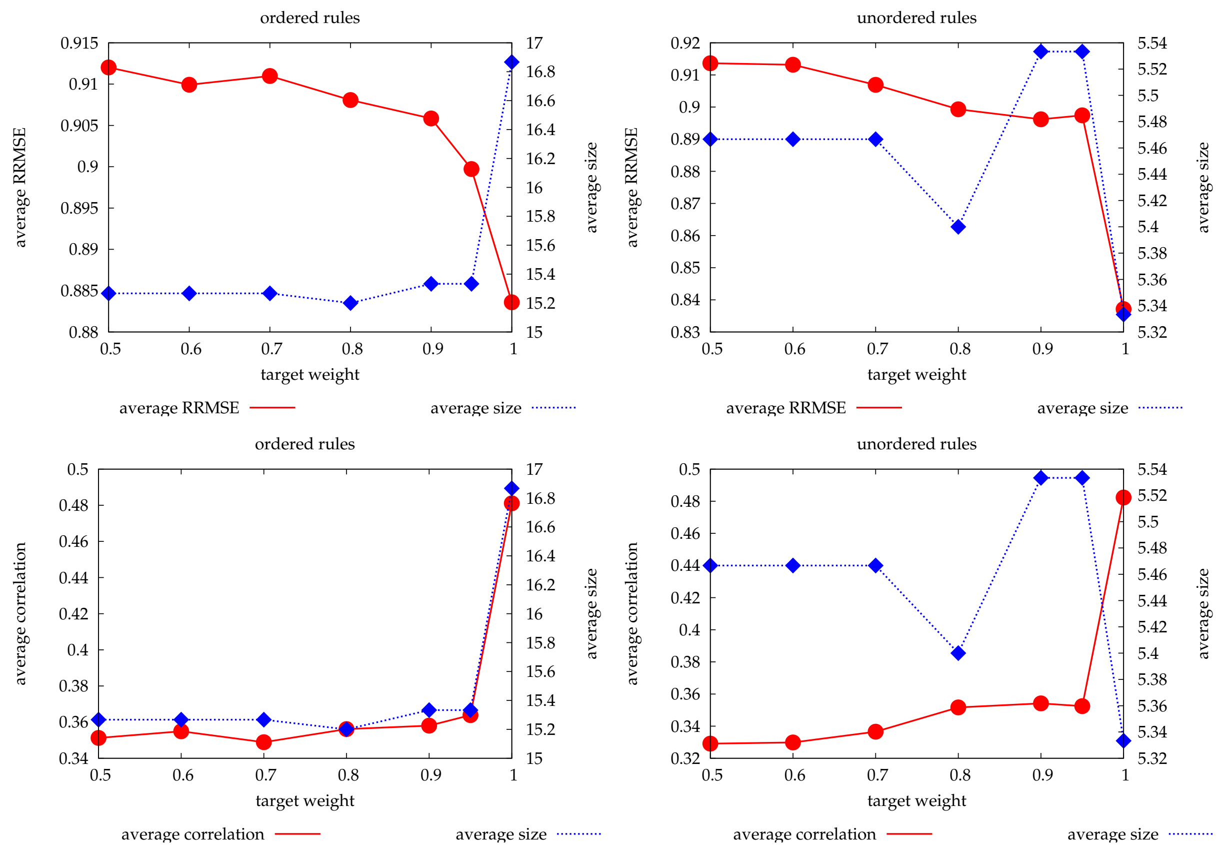


Figure 5.14: *Single target regression problems,* influence of target attribute weights on average RRMSE, correlation coefficients, and rule set size for ordered and unordered PCRs. Averages are computed over all single target regression data sets.

attributes. In this section we empirically investigate the influence of this parameter for each of the four types of problem domains; we have been changing the value of the parameter between 0.5 and 1.

Single target classification problems. The target weight parameter influence is illustrated in Figure 5.13, separately for ordered and unordered rules. The figure shows average error rates and rule set sizes over all single target classification data sets. The results are very similar for ordered and unordered rules. As expected, the parameter value of one, i.e., no influence of the descriptive attributes, gives rise to rule sets with the lowest average error rate. In fact, even a slight influence of the descriptive attributes causes a large increase of average error rates. As for the size of the rule sets, we can conclude that a moderate influence of descriptive attributes within the search heuristic tends to decrease the number of learned rule sets, while even larger influence increases the number again. This regularity is somewhat broken in the case of unordered rules, because rule sets with the parameter set to 1 are on aver-

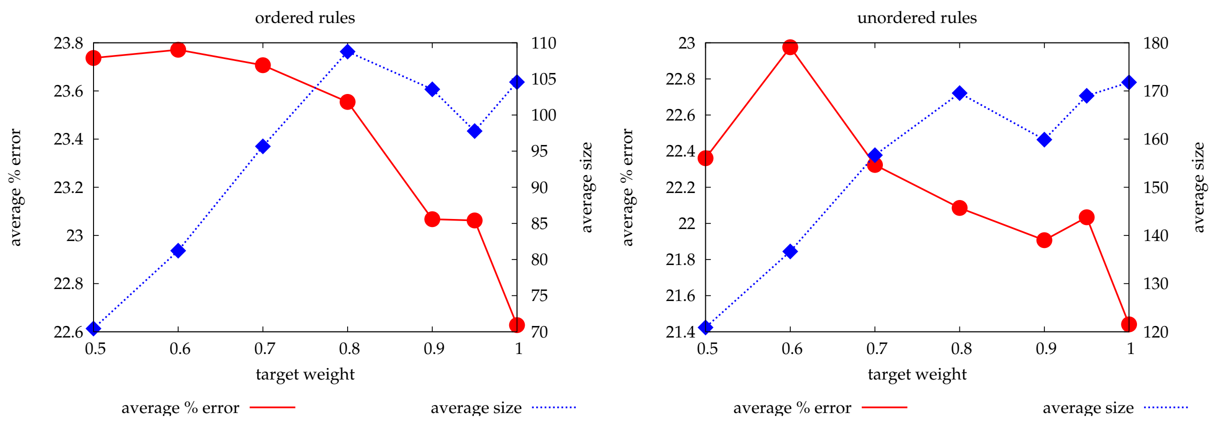


Figure 5.15: *Multiple target classification problems,* influence of target attribute weights on average error rate and rule set size for ordered and unordered PCRs. Averages are computed over all target attributes of all multiple target classification data sets.

age the smallest and most accurate. The initial drop of the number of rules could be explained by a more cautious specialization of rules, since the non-target attributes represent additional constraints to homogeneity of examples covered by a rule. In addition, irrelevant non-target attributes can hinder the learning of accurate rules.

Single target regression problems. The target weight parameter influence is illustrated in Figure 5.14, separately for ordered and unordered rules, and separately for RRMSE and the correlation coefficient. The figure shows the average RRMSE, correlation coefficient, and rule set sizes over all single target regression data sets. When considering the accuracy (RRMSE and correlation coefficient), the results are the same as in the case of single target classification. Even a slight influence of the descriptive attributes drastically increases the RRMSE and decreases the correlation coefficient. As in the case of single target classification, parameter values of less than one decrease the size of ordered PCRs. The differences in sizes of unordered rules are very small, however, the trend is similar as in the case of unordered classification rules.

Multiple target classification problems. The target weight parameter influence is illustrated in Figure 5.15, separately for ordered and unordered rules. The figure shows average error rates over all target attributes of all multiple target classification data sets, and average rule set sizes over all multiple target classification data sets. The results are, again, very similar for ordered and unordered rules, and also similar to the single target classification case (with the exception of the unordered rule sets at τ=1 being the smallest). The larger the influence of the non-target (descriptive) attributes, the larger the average error rate, and smaller average rule set size. The

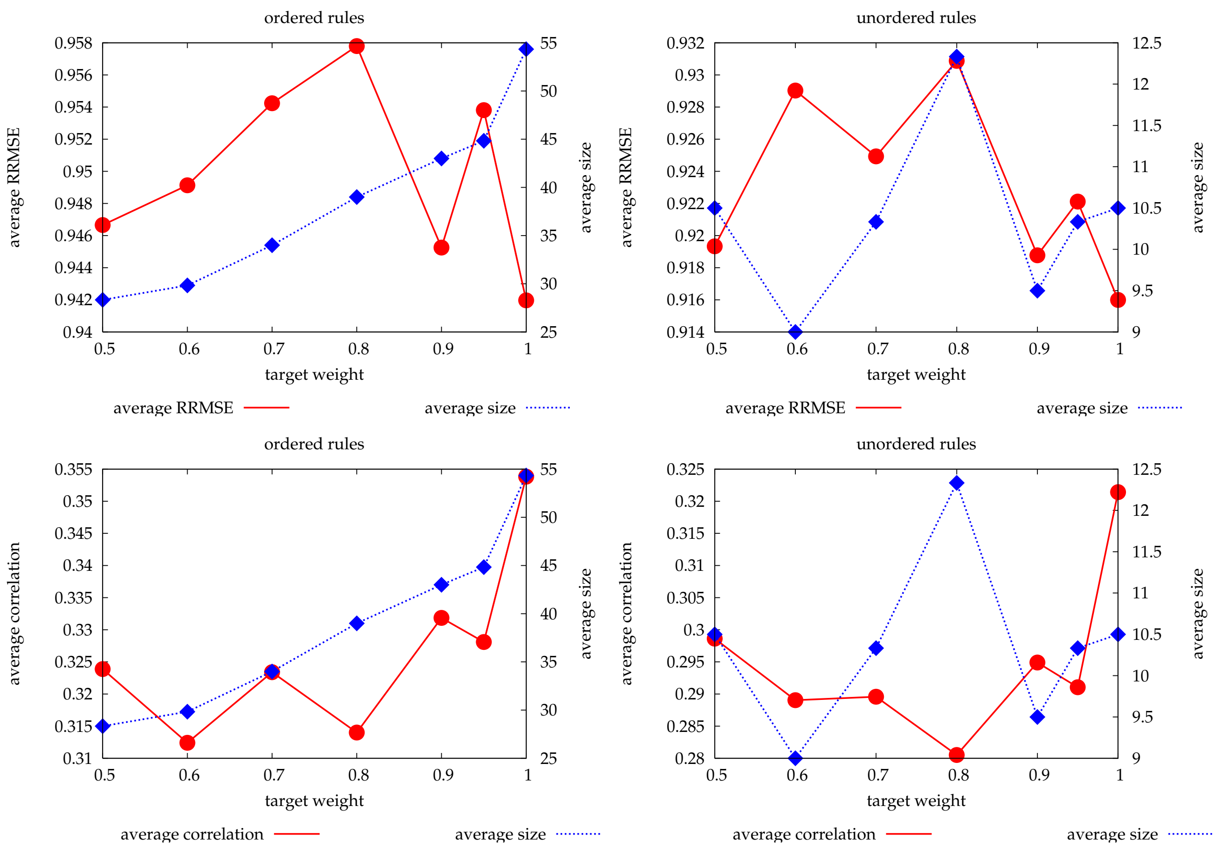


Figure 5.16: *Multiple target regression problems,* influence of target attribute weights on average RRMSE, correlation coefficients, and rule set size for ordered and unordered PCRs. Averages are computed over all target attributes of all multiple target regression data sets.

trends, however, are not completely monotonic.

Multiple target regression problems. The target weight parameter influence is illustrated in Figure 5.16, separately for ordered and unordered rules, and separately for RRMSE and the correlation coefficient. The figure shows the average RRMSE, correlation coefficient, and rule set sizes over all target attributes of all multiple target regression data sets. Again, the results are similar as in the case of single target regression. The larger the influence of the non-target (descriptive) attributes, the larger the average RRMSE, and the smaller the correlation coefficient. The influence on the rule set size of ordered rules very similar as in all previous cases. The trend in the unordered rule set sizes is less obvious, though some similarities with other unordered rules exist, e.g., the rule sets at τ=1 tend to be rather small.

The presented results suggest that, as expected, the target weight parameter must be set to the value of one (the search heuristic should not take into account the descriptive attributes), in order to get the best predictive accuracy. The parameter's influence on the rule set size is somewhat different for ordered and unordered rules. In the case of ordered rules, the rule set size decreases with inclusion of the descriptive attributes in the search heuristic. The size of unordered rule sets, however, with the decreasing of the τ sometimes briefly peaks before decreasing as in the case of ordered rules. The drop of the number of rules can be explained by a more cautious specialization of rules, since the non-target attributes represent additional constraints to rule learning, and by the influence of irrelevant non-target attributes.

5.5 Influence of covering weight

Unordered predictive clustering rules (PCRs) are learned using the error weighted covering algorithm (see Section 4.5). When this algorithm learns a rule, it modifies the current learning set, namely, it reduces the weights of examples covered by this rule. The amount by which the weights are reduced is inversely proportional to the error the rule makes when predicting the values of target attributes for each example, and to the *covering weight parameter* ζ (see Equations 4.6 and 4.8). It should be set to a value between (or equal to) zero and one. If its value is set to zero, this means that all examples covered by a new rule are removed from the learning set (since their weight is set to zero); this is the standard (non-weighted) covering algorithm. In principle, a larger value of the parameter should keep the examples in the learning set longer, which should result in more iterations of the learning process, which in turn, should lead to a larger number of learned rules. In this section, we empirically investigate the influence of this parameter for each of the four types of problem domains. Based on some preliminary experiments, we have decided to investigate values of the parameter between 0 and 0.3.

Single target classification problems. The covering weight parameter influence is illustrated in Figure 5.17. The figure shows average error rates and rule set sizes over all single target classification data sets. The graph shows that setting the parameter value from 0 (or 0.01) to 0.1 drastically reduces the average error rate. This drop of error rate is combined with an increase of the average rule set size. Additional increase of the parameter value causes further increase of the rule set size, but does not reduce the error rate. The results suggest that the covering weight parameter value of 0.1 is a reasonable compromise between the rule set's error rate and its size for the single target classification domains, though the actual optimal value may be domain dependent.

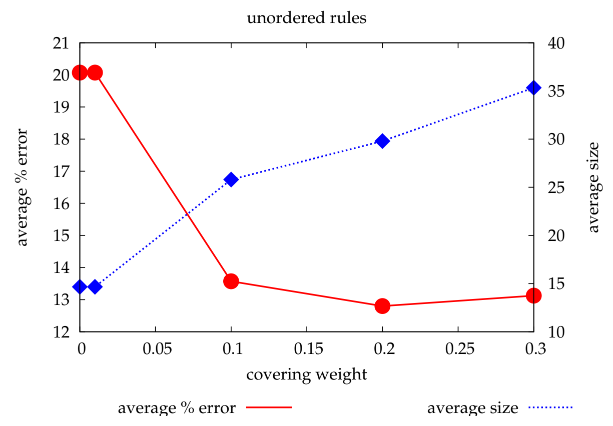


Figure 5.17: *Single target classification problems,* influence of the covering weight parameter on error rate and rule set size of unordered PCRs. Averages are computed over all single target classification data sets.

Single target regression problems. The covering weight parameter influence is illustrated in Figure 5.18, separately for RRMSE and correlation coefficient. The figure shows average RRMSE, correlation coefficient, and rule set sizes over all single target regression data sets. The results are showing the opposite picture as compared to the case of classification. The increase of the parameter value increases the average RRMSE and decreases the average correlation coefficient and the average rule set size. This suggest that most accurate unordered PCRs can be learned by the standard covering algorithm, i.e., by setting the covering weight parameter to the value of 0.

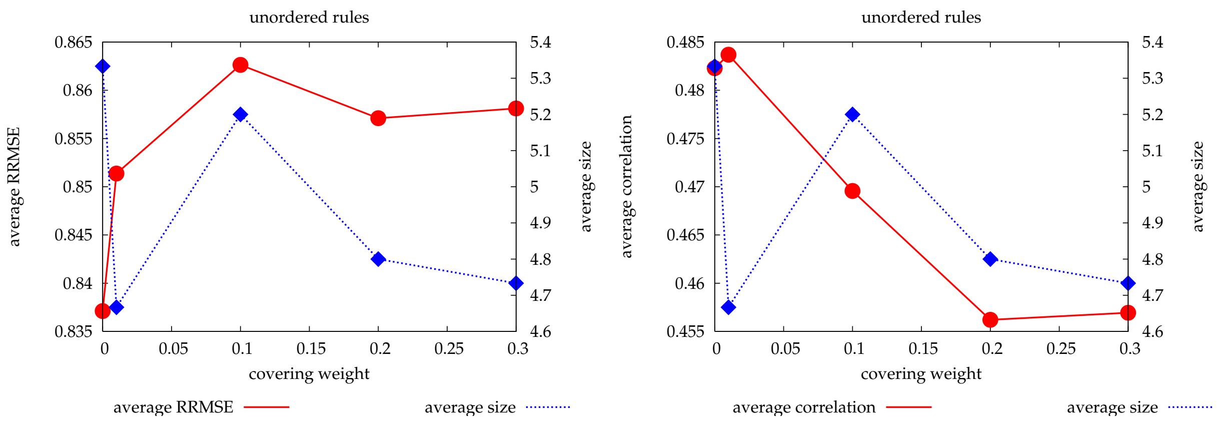


Figure 5.18: *Single target regression problems,* influence of the covering weight parameter on RRMSE, correlation coefficients, and rule set size of unordered PCRs. Averages are computed over all single target regression data sets.

Multiple target classification problems. The covering weight parameter influence is illustrated in Figure 5.19. The figure shows average error rates over all target attributes of all multiple target classification data sets, and average rule set sizes over all multiple target classification data sets. The results are very similar to the single target classification case. By increasing the parameter value the average error rate drops, while the average rule set size increases. This results suggest that the optimal values for the covering weight parameter are between 0.1 and 0.2.

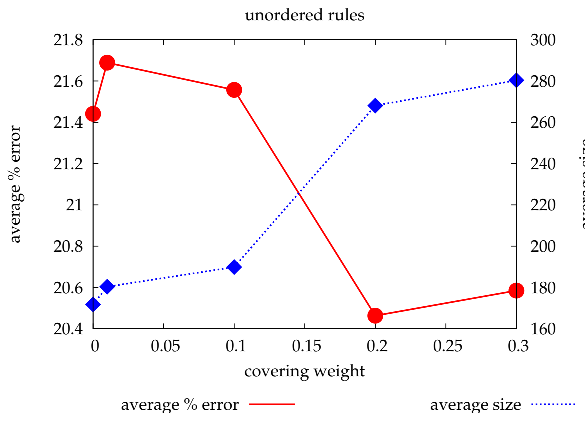


Figure 5.19: *Multiple target classification problems*, influence of the covering weight parameter on error rate and rule set size of unordered PCRs. Averages are computed over all target attributes of all multiple target classification data sets.

Multiple target regression problems. The covering weight parameter influence is illustrated in Figure 5.20, separately for RRMSE and correlation coefficient. The figure shows average RRMSE, correlation coefficient, and rule set sizes over all single target regression data sets. Again, the results are showing a rather different picture as compared to the case of classification, and are similar to those for single target regression. With some exceptions, the increase of the parameter value tends to increase the average RRMSE and decrease the average correlation coefficient and the average rule set size.

The presented results suggest that for classification problems, the covering weight parameter can be used to balance between the accuracy of the learned rule sets and their size. For the regression problems, however, setting the parameter to a very low value seems work best. We believe the main reason for different influence of the parameter in classification and in regression tasks is the fact that it is defined in a somewhat different way for each of the tasks (Equations 4.6 and 4.8).

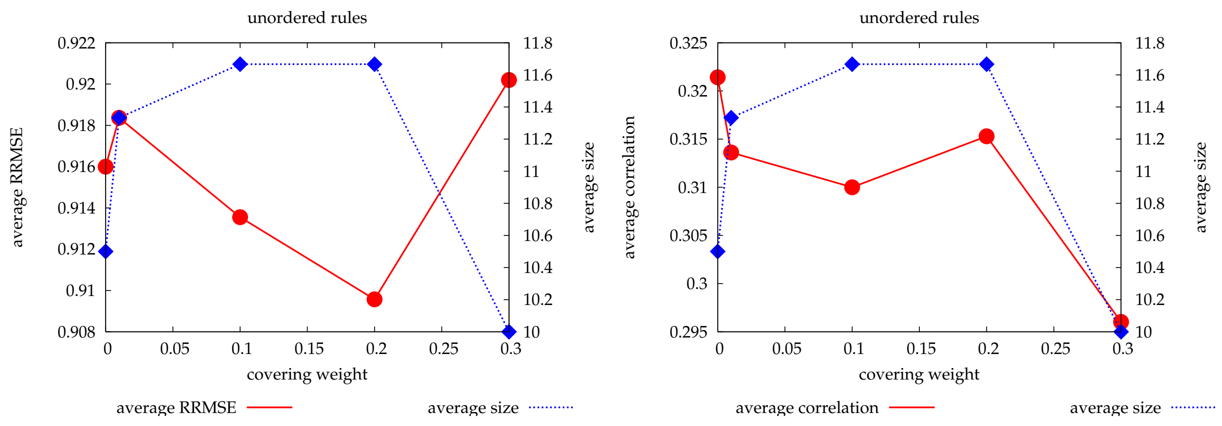


Figure 5.20: *Multiple target regression problems,* influence of the covering weight parameter on RRMSE, correlation coefficients, and rule set size of unordered PCRs. Averages are computed over all target attributes of all multiple target regression data sets.

5.6 Summary

In this chapter, the empirical evaluation of the the newly developed methods for induction of predictive clustering rules (PCRs) is presented. The performed experiments compare PCRs to some existing approaches, compare multiple target prediction to single target prediction, and investigate the influence of some learning algorithm parameters. Let us briefly summarize the results.

Comparison to existing methods. The experiments on single target classification problems show that the performance of ordered predictive clustering rules (PCRs) is not significantly different from that of ordered rules learned with CN2 and CN2-WRAcc methods. Ordered PCRs are also not significantly different from that of predictive clustering trees (PCTs) (though somewhat worse). Ordered JRip rules are also somewhat better than ordered PCRs (*p-value*=0.07). Unordered PCRs are better than unordered CN2 and CN2-WRAcc rules, due to the error weighted covering algorithm used for learning unordered PCRs. The unordered PCRs are not significantly different from unordered CN2-EVC and ordered JRip rules, and trees (PCTs), though somewhat better. Unordered PCRs are better than ordered PCRs.

The results of the evaluation on single target regression problems suggest that both ordered and unordered PCRs are better than (ordered) FRS rules. The comparison to trees (PCTs), however, shows that PCRs are significantly worse than trees.

The evaluation on multiple target classification problems shows that ordered PCRs are somewhat worse than trees (PCTs), but not significantly. In addition they tend to produce a smaller number of rules than (transcribed) trees. The same holds true for unordered PCRs, but this time PCRs are (not significantly) better than trees.

The experiments on multiple target regression problems and the comparison of PCRs to trees show a similar picture as in the case of single target regression. Ordered, as well as unordered, PCRs are significantly worse than trees (PCTs). We believe the main reason that PCTs are better than PCRs on single and multiple target regression problems is the fact that PCTs use the state-of-the-art post-pruning method, while PCRs use no post-pruning.

Comparison of single target and multiple target prediction. The comparison of multiple target prediction PCRs to the corresponding sets of single target prediction PCRs on classification problems shows that in the case of ordered rules, the single target prediction models are better, while in the case of unordered rules, the multiple target prediction PCRs are better. The differences in both cases are almost (but not quite) significant. The difference in the rule set sizes, on the other hand, is very significant. Multiple target prediction ordered and unordered rule sets are much smaller than the corresponding single target prediction rule sets.

The results of the evaluation on the multiple target regression problems are somewhat contradicting: we believe there is not enough evidence to conclude that the accuracy of single and multiple target PCRs are significantly different. The sizes of the multiple target PCRs, on the other hand, are again significantly smaller.

Influence of target attribute weights. Adding weight to the non-target (descriptive) attributes decreases the accuracy of the learned PCR rule sets. It also generally decreases the rule set size, though in the case of unordered rules there are some exceptions.

Influence of covering weight. For classification problems, the covering weight parameter can be used to balance between the accuracy of the learned rule sets and their size. The larger the parameter's value, the better the accuracy, but also the larger rule set size. For the regression problems, however, setting the parameter to a very low value produces the most accurate rule sets.

Chapter 6

Conclusions

In this thesis we developed and empirically evaluated a method for learning predictive clustering rules. The method combines ideas from supervised and unsupervised learning and extends the predictive clustering approach to methods for rule learning. In addition, it generalizes rule learning and clustering. The newly developed algorithm is empirically evaluated, in terms of performance, on several single and multiple target classification and regression problems. The new method compares favorably to existing methods. The comparison of single target and multiple target prediction models shows that multiple target models offer comparable performance and drastically lower complexity than the corresponding sets of single target models.

6.1 Original contributions

The work presented in this thesis comprises several contributions to the area of machine learning. First, we have developed a new method for learning unordered single target classification rules. It is loosely based on the commonly used rule learning method CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), but uses a generalized *weighted covering* algorithm (Gamberger and Lavrač, 2002).

Second, the developed method is generalized for learning ordered or unordered rules, on single or multiple target classification or regression domains. It uses a search heuristic that takes into account several rule quality measures and is applicable to all the above mentioned types of domains.

The third contribution of the thesis is the extension of the predictive clustering approach to models in the form of rules. The newly developed method combines rule learning and clustering. The search heuristic takes into account the values of both the target and the descriptive attributes. Different weighting of these two types of attributes enable us to traverse from predictive modeling to clustering.

The final contribution is an extensive empirical evaluation of the newly developed method on single target classification and regression problems, as well as multiple target classification and regression problems. Performance of the new method is compared to some existing methods. The results show that on single target classification problems, the performance of predictive clustering rules (PCRs) is comparable to that of CN2 rules and predictive clustering trees (PCTs), while in the case of unordered rules, PCRs are better than CN2 rules. Unordered PCRs are in general better than ordered PCRs. On multiple target classification problems, PCRs are comparable to PCTs, but PCRs tend to produce smaller rule sets than (transcribed) trees. Single target regression PCRs are comparable to FRS rules, however, their performance is much worse than that of PCTs; on multiple target regression problems, PCRs are also much worse than PCTs. We believe the main reason that PCTs are better than PCRs on regression problems is the fact that PCTs use a state-of-the-art post-pruning method, while PCRs use no post-pruning. The comparison of the performance of single target and multiple target PCRs on multiple target problems shows, that multiple target prediction provides comparable accuracy as single target prediction, but multiple target prediction rule sets are much smaller than the corresponding single target rule sets.

6.2 Further work

Let us conclude with some guidelines for further work. The thesis proposes a relatively general algorithm for rule learning which covers several possible methods for rule learning. We were only able to investigate and evaluate the basic covering approach to predictive clustering rules. The algorithm includes several other rule learning approaches, which have not yet been evaluated. These include the 'distance to existing rules' and 'prototype dissimilarity' parts of the heuristic search function (Section 4.3, Equations 4.1 and 4.2), *"Sampling"* and *"None"* learning set modifying methods (Section 4.5, Table 4.4), and optimization of rule weights, i.e., the *"Unordered-Opt-W"* prediction method (Section 4.6, Table 4.6).

In addition, there exist several newer methods, e.g., Ripper (Cohen, 1995), Slipper (Cohen and Singer, 1999), and CN2-EVC (Možina et al., 2006), which tend to give better results on single target classification problems. Incorporating ideas from these methods into predictive clustering rules could lead to improved performance.

The evaluation showed that the performance of current regression rules methods based on the covering approach is significantly worse than the performance of regression trees, and that a further development of regression rules learning algorithms is necessary. We believe the inclusion of reduced error pruning (Brunk and Pazzani,

1991) into the PCR algorithm would significantly improve its performance. Additionally, introduction of prototypes in the form of (linear) regression models into PCRs should also lead to increased performance of regression rules.

The search heuristic used in our algorithm is multi-objective. It comprises several criteria and we have combined these in a relatively simple manner. Employing approaches developed within the decision support (Keeney and Raiffa, 1993) and multi-objective optimization (Deb, 2001) areas could lead to improved performance of predictive clustering rules.

Bibliography

H. Blockeel. *Top-down Induction of First Order Logical Decision Trees*. PhD thesis, Katholieke Universiteit Leuven, Department of Computer Science, Leuven, Belgium, 1998. 3, 10, 11, 12, 66, 121, 122

H. Blockeel and J. Struyf. Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research*, 3:621–650, December 2002. 59, 66

H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In J. W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 98)*, pages 55–63, San Francisco, CA, USA, July 1998. Morgan Kaufmann. 3, 10, 35, 59, 66, 121, 122

M. Bohanec and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994. 62

I. Bratko. *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Pearson Education, Harlow, UK, 3rd edition, 2001. 1

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 43

C. Brunk and M. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. In L. Birnbaum and G. Collins, editors, *Proceedings of the Eighth International Workshop on Machine Learning*, pages 389–393, San Francisco, CA, USA, 1991. Morgan Kaufmann. 76, 106

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. 12

B. Cestnik. Estimating probabilities: A crucial task in machine learning. In L. Aiello, editor, *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI 90)*, pages 147–149, London, UK/Boston, MA, USA, 1990. Pitman. 13, 26

P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, pages 151–163, Berlin, Germany, 1991. Springer. 4, 13, 24, 36, 40, 42, 44, 46, 59, 105, 122, 126, 127, 130

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989. 4, 13, 24, 35, 40, 42, 44, 45, 59, 105, 122, 126, 127, 130

W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, San Francisco, CA, USA, 1995. Morgan Kaufmann. 59, 76, 106

W. W. Cohen and Y. Singer. A simple, fast, and effictive rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 99)*, pages 335–342, Menlo Park, CA, USA, 1999. AAAI Press/MIT Press. 106

N. Colbach, C. Clermont-Dauphin, and J. M. Meynard. GeneSys: A model of the influence of cropping system on gene escape from herbicide tolerant rapeseed crops to rape volunteers - I. temporal evolution of a population of rapeseed volunteers in a field. *Agriculture, Ecosystems and Environment*, 83(3):232–253, 2001a. 57

N. Colbach, C. Clermont-Dauphin, and J. M. Meynard. GeneSys: A model of the influence of cropping system on gene escape from herbicide tolerant rapeseed crops to rape volunteers - II. genetic exchanges among volunteer and cropped populations in a small region. *Agriculture, Ecosystems and Environment*, 83(3):255–270, 2001b. 57

T. Curk, U. Petrovič, G. Shaulsky, and B. Zupan. Rule-based clustering for gene regulation pattern discovery. In *Proceedings of the Workshop on Intelligent Data Analysis in Biomedicine and Pharmacology (IDAMAP 2006)*, pages 45–50, Verona, Italy, 2006. Department of Computer Science, University of Verona. 12

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, New York, NY, USA, 2001. 107

D. Demšar. Obravnavanje numeričnih problemov z induktivnim logičnim programiranjem. Masters thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, 1999. In Slovene. 59, 64

D. Demšar, M. Debeljak, C. Lavigne, and S. Džeroski. Modelling pollen dispersal of genetically modified oilseed rape within the field. In *Abstracts, 90th ESA Annual Meeting*, page 152, Montreal, Canada, 2005. The Ecological Society of America. 57

D. Demšar, S. Džeroski, T. Larsen, J. Struyf, J. Axelsen, M. B. Pedersen, and P. H. Krogh. Using multi-objective classification to model communities of soil microarthropods. *Ecological Modelling*, 191(1):131–143, 2006. 56

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006. 26, 49, 60, 61, 79

J. Demšar, B. Zupan, and G. Leban. Orange: From experimental machine learning to interactive data mining, White paper. Faculty of computer and information science, University of Ljubljana, 2004. URL www.ailab.si/orange. 60

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973. 9

S. Džeroski, B. Cestnik, and I. Petrovski. Using the m-estimate in rule induction. *Journal of Computing and Information Technology*, 1(1):37–46, 1993. 14

S. Džeroski, D. Demšar, and J. Grbović. Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, 13(1):7–17, 2000. 58

S. Džeroski, N. Colbach, and A. Messéan. Analysing the effect of field character on gene flow between oilseed rape varieties and volunteers with regression trees. In A. Messéan, editor, *Proceedings of the Second International Conference on Co-existence between GM and non-GM based Agricultural Supply Chains*, pages 207–211, Montpellier, France, November 2005. Agropolis Productions. 58

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, May 1994. 43

P. Flach and N. Lavrač. Rule induction. In M. R. Berthold and D. J. Hand, editors, *Intelligent Data Analysis*, pages 229–267, Berlin, Germany, 2003. Springer. Second edition. 2, 8, 12, 120, 122

J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999. 16

J. H. Friedman and B. E. Popescu. Gradient directed regularization. Technical report, Stanford University, Stanford, CA, USA, 2004. 46, 47

J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. Technical report, Stanford University, Stanford, CA, USA, 2005. 17, 35, 46, 47

J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1): 3–54, February 1999. 3

J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In W. W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning*, pages 70–77, San Francisco, CA, USA, 1994. Morgan Kaufmann. 40

D. Gamberger and N. Lavrač. Expert guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002. 4, 15, 42, 105, 127, 130

T. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, Berlin, Germany, 2001. 7, 9, 25, 120

G. Holmes, M. Hall, and E. Frank. Generating rule sets from model trees. In *Proceedings of the Twelfth Australian Joint Conference on Artificial Intelligence (AI 99)*, pages 1–12, London, UK, 1999. Springer. 17, 35, 59

A. Karalič and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997. 16, 56, 59, 64, 123

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, NY, USA, 1990. 2, 9, 121

R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge University Press, Cambridge, UK, 1993. 107

W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In U. M. Fayyad, G. Piatetsky-Shapiro, and R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 249–271, Menlo Park, CA, USA, 1996. AAAI Press/MIT Press. 15

I. Kononenko. *Strojno učenje*. Založba FE in FRI, Ljubljana, Slovenia, 2005. In Slovene. 25

I. Kononenko and I. Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6(1):67–80, 1991. 24

P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 1996. 1, 3, 10, 121

N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In S. Džeroski and P. Flach, editors, *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP 99)*, Lecture Notes in Artificial Intelligence, pages 174–185, Berlin, Germany, 1999. Springer. 14, 38, 126, 127

N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004. 15, 46, 123

R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140: 5–53, 1932. 54

R. S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing (FCIP 69)*, volume A3, Switching Circuits, pages 125–128, Bled, Yugoslavia, 1969. 3, 13, 33, 41, 122, 125

R. S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, 4:219–243, 1980. 2, 10, 121

R. S. Michalski, I. Mozetič, J. Hong, and N. Lavrač. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI 86)*, pages 1041–1047, Philadelphia, PA, USA, 1986. Morgan Kaufmann. 13, 33, 41, 122, 125

T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997. 1, 12, 122

M. Možina, J. Demšar, J. Žabkar, and I. Bratko. Why is rule learning optimistic and how to correct it. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006, Proceedings of the Seventeenth European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006*, Lecture Notes in Computer Science, pages 330–340. Springer, 2006. 26, 59, 106

P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, Princeton, NY, USA, 1963. 59, 62, 79

D. Newman, S. Hettich, C. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. URL http://www.ics.uci.edu/~mlearn/MLRepository.html. 51, 52, 128

K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, Berlin, Germany, 2005. 47

J. R. Quinlan. Learning with continuous classes. In A. Adams and L. Sterling, editors, *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence, Hobart, Australia, November 16-18, 1992*, pages 343–348, Singapore, 1992. World Scientific. 76

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993. ISBN 1-55860-238-0. 2, 8, 11, 120

J. R. Quinlan. MDL and categorial theories (continued). In A. Prieditis and S. Russel, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 464–470, San Francisco, CA, USA, 1995. Morgan Kaufmann. 17, 35

M. Škrjanc, M. Grobelnik, and D. Zupanič. Insights offered by data-mining when analyzing media space data. *Informatica*, 25(3):357–363, 2001. 53

M. Šprogar, P. Kokol, Š. H. Babič, V. Podgorelec, and M. Zorman. Vector decision trees. *Intelligent Data Analysis*, 4(3-4):305–321, 2000. 12

StatLib. Data sets archive. URL http://stat.cmu.edu/datasets/. 52

S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. D. Jong, S. Džeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. V. de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CS-91-197, Carnegie Mellon University, Pittsburgh, PA, USA, 1991. 54

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. 47

L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In D. A. Zighed, J. Komorowski, and J. Żytkow, editors, *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2000)*, Lecture Notes in Computer Science, pages 255–264, Berlin, Germany, 2000. Springer. 14, 38, 59

L. Torgo. Data fitting with rule-based regression. In J. Žižka and P. Brazdil, editors, *Proceedings of the Second International Workshop on Artificial Intelligence Techniques (AIT 95)*, Brno, Czech Republic, 1995. 15, 123

L. Torgo and J. Gama. Regression by classification. In D. Borges and C. Kaestner, editors, *Proceedings of the Brazilian Symposium on Artificial Intelligence (SBIA 96)*, Lecture Notes in Artificial Intelligence, pages 51–60, Berlin, Germany, 1996. Springer. 15

S. M. Weiss and N. Indurkhya. Rule-based regression. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 93), Chambéry,* *France, August 28-September 3, 1993*, pages 1072–1078, San Francisco, CA, USA, 1993. Morgan Kaufmann. 15, 59

S. M. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995. 15

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945. 49, 59, 79, 128

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, USA, second edition, 2005. 60

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. 9

B. Ženko, S. Džeroski, and J. Struyf. Learning predictive clustering rules. In F. Bonchi and J.-F. Boulicaut, editors, *Fourth International Workshop on Knowledge Discovery in Inductive Databases (KDID 2005), revised selected and invited papers*, Lecture Notes in Computer Science, pages 110–121, Berlin, Germany, 2006. Springer. 33

Appendices

Dodatek A

Razširjeni povzetek Učenje pravil za napovedno razvrščanje

Učenje pravil za napovedno razvrščanje, predstavljeno v pričujoči disertaciji, temelji na idejah z dveh področij strojnega učenja, napovednega modeliranja in razvrščanja v skupine. Omenjeni področji sta običajno obravnavani popolnoma ločeno. Napovedno modeliranje se ukvarja z gradnjo modelov za napovedovanje določenih ciljnih lastnosti objektov v odvisnosti od opisov teh objektov, razvrščanje pa se po drugi strani ukvarja z razvrščanjem objektov v skupine medseboj podobnih si objektov. V tem primeru nimamo ciljnih lastnosti, ki naj bi jih napovedovali, in navadno tudi ne simboličnega opisa zgrajenih skupin objektov. Kljub temu pa je med metodami za napovedno modeliranje, ki delijo prostor primerov, kot so na primer odločitvena drevesa in pravila, ter razvrščanjem v skupine tudi nekaj podobnosti. Omenjene metode napovednega modeliranja delijo prostor primerov v podskupine primerov s čim bolj podobnimi vrednostmi ciljne spremenljivke, medtem ko razvrščanje išče skupine primerov, ki so si čim bolj podobni v vrednostih vseh opisnih spremenljivk. Napovedno razvrščanje temelji na tej podobnosti. Išče skupine primerov, ki so si čim bolj podobni, pri tem pa upošteva tako opisne kot tudi ciljne spremenljivke. Ob tem vsaki skupini primerov določi tudi napovedni model. Metode napovednega razvrščanja nam omogočajo gradnjo modelov za napovedovanje več ciljnih spremenljivk hkrati. Taki modeli so običajno preprostejši in bolj razumljivi kot ustrezna množica modelov za napovedovanje zgolj ene ciljne spremenljivke.

Doslej je bilo napovedno razvrščanje omejeno na metode za gradnjo odločitvenih dreves. Cilj te disertacije je razširiti pristop napovednega razvrščanja na metode za učenje pravil oz. razviti metodo, ki bo obravnavala učenje pravil in razvrščanje v skupine na enoten način. Večina obstoječih metod za učenje pravil temelji na prekrivnem algoritmu, ki je bil prvotno zasnovan za učenje odločitvenih seznamov pravil za binarne klasifikacijske probleme. Razvili smo posplošeno verzijo tega algoritma, ki nam omogoča učenje urejenih in neurejenih množic pravil na klasifikacijskih ali regresijskih problemih z eno ali več ciljnimi spremenljivkami.

Novo razvito metodo smo empirično ovrednotili na množici problemov z eno ali več ciljnimi spremenljivkami. Rezultati so primerljivi z že obstoječimi metodami za učenje pravil in drevesi za napovedno razvrščanje. Primerjava modelov za napovedovanje več ciljnih spremenljivk hkrati z modeli za napovedovanje posameznih ciljnih spremenljivk pokaže, da prvi ponujajo podobno ali boljšo točnost napovedi, ob tem pa vsebujejo bistveno manj pravil.

A.1 Izhodišča in obstoječe metode

Področje pričujoče disertacije je učenje pravil za napovedno razvrščanje in je povezana z več področji strojnega učenja. Napovedno razvrščanje združuje pristope napovednega modeliranja in razvrščanja v skupine. V nadaljevanju bomo na kratko predstavili napovedno modeliranje, razvrščanje v skupine in napovedno razvrščanje. Učenje pravil je sicer podpodročje napovednega modeliranja, vendar mu bomo zaradi pomembnosti za pričujočo disertacijo namenili poseben razdelek. Vsako od omenjenih področij bomo na kratko predstavili ter omenili raziskave povezane s pričujočo disertacijo.

A.1.1 Napovedno modeliranje

Napovedno modeliranje (Hastie et al., 2001) se v splošnem ukvarja z gradnjo modelov, ki nam na osnovi opisa nekega objekta omogočajo napovedovanje določene ciljne lastnosti tega objekta. Napovedni model gradimo, oziroma se ga učimo, na osnovi učne množice objektov ali učnih primerov. Opis objekta je največkrat podan kot vektor atributnih vrednosti, čeprav se uporabljajo tudi drugi opisni jeziki, kot na primer logika prvega reda. Ciljna lastnost objekta je običajno predstavljena z eno samo ciljno spremenljivko, ki jo imenujemo razred. V kolikor je ta spremenljivka nominalna, govorimo o *klasifikacijskem problemu*, če je numerična oziroma zvezna, pa o *regresijskem problemu*. Napovedni model, ki ga dobimo kot rezultat učenja, je lahko predstavljen v veliko različnih oblikah, od enačb do logičnih programov. Različne oblike modelov se med drugim razlikujejo tudi po napovedni točnosti in razumljivosti. Med razumljivimi oblikami modelov se poleg enačb zelo pogosto uporabljajo tudi odločitvena drevesa (Quinlan, 1993) in pravila (Flach in Lavrač, 2003). Za razliko od enačb, ki napovedujejo vrednost ciljne spremenljivke v celotnem prostoru primerov, drevesa in pravila ta prostor najprej razdelijo, nato pa za vsak podprostor zgradijo poseben (bolj preprost) model. Napovedni modeli v obliki pravil so tema pričujoče disertacije.

A.1.2 Razvrščanje v skupine

Razvrščanje v skupine (Kaufman in Rousseeuw, 1990) se v splošnem ukvarja z razvrščanjem objektov v skupine ali razrede podobnih objektov. Podano imamo množico opisov objektov oziroma primerov, ki jih želimo razvrstiti v razrede tako, da so si posamezni primeri v isti skupini med seboj čim bolj podobni ter hkrati, da so si primeri v različnih skupinah med seboj čim bolj različni. Primeri so, podobno kot pri napovednem modeliranju, običajno opisani z množico vrednosti neodvisnih spremenljivk ali atributov, vendar pa nimajo določene ciljne lastnosti, ki bi jo napovedovali. Za razvrščanje potrebujemo neko mero podobnosti, s katero določamo razdalje med posameznimi objekti. Objekti so predstavljeni kot točke v večdimenzionalnem metričnem prostoru, v katerem so razdalje med posameznimi objekti enoznačno določene. Skupino objektov lahko predstavimo s prototipnim objektom, ki je na primer točka v prostoru z najmanjšo povprečno razdaljo do vseh elementov skupine. Na ta način lahko računamo tudi razdalje med skupinami objektov. Pri računanju razdalj moramo biti pozorni na normalizacijo posameznih atributov oziroma nijhovo uteževanje, ker lahko vsak atribut predstavlja različno meritev, ki je lahko podana v različnih enotah. Končni rezultat razvrščanja so skupine objektov, ne pa tudi njihovi simbolni opisi. Kljub temu lahko na zgrajene skupine gledamo kot na razrede pri klasifikaciji ter naknadno zgradimo simbolne opise že zgrajenih skupin. Ta pristop imenujemo *konceptualno razvrščanje* (Michalski, 1980). Za razvrščanje lahko uporabimo tudi odločitvena drevesa. Vsako vozlišče drevesa obravnavamo kot skupino in celotno drevo nam predstavlja hierarhično urejene skupine. Takemu drevesu pravimo *razvrščevalno drevo* (Blockeel, 1998) in vsaka skupina v njem je opisana s konjunkcijo pogojev, ki se nahajajo na poti od korena drevesa do danega vozlišča. Skupine v razvrščevalnem drevesu, ki niso v isti veji, se med seboj ne prekrivajo.

A.1.3 Napovedno razvrščanje

Na podlagi povedanega lahko sklepamo, da so metode napovednega modeliranja, ki delujejo na principu deljenja množice primerov v podmnožice, v bistvu zelo podobne metodam za razvrščanje (Langley, 1996). Glavna razlika je v tem, da prve delijo primere v skupine, ki imajo homogene vrednosti ciljne spremenljivke, razvrščanje pa deli primere v skupine, ki imajo homogene vrednosti vseh opisnih spremenljivk. Na osnovi te podobnosti je nastala ideja *napovednega razvrščanja* (Blockeel, 1998; Blockeel et al., 1998), ki združuje oba pristopa. Pri napovednem razvrščanju, enako kot pri običajnem razvrščanju, iščemo skupine medseboj podobnih si elementov, vendar pri tem upoštevamo tako opisne (neodvisne) spremenljivke, kot tudi ciljne (odvisne) spremenljivke. Poleg tega vsaki skupini pripišemo model, ki to skupino opisuje in hkrati na osnovi vrednosti opisnih spremenljivk napoveduje vrednosti ciljnih spremenljivk. Z metodami napovednega razvrščanja lahko gradimo modele za napovedovanje več spremenljivk hkrati, ki so velikokrat bolj enostavni in bolj razumljivi kot ustrezno število modelov za napovedovanje posameznih spremenljivk. Ker iskanje podobnih primerov poteka ob upoštevanju vseh atributov, dosega napovedno razvrščanje posebej dobre rezultate na domenah z veliko šuma ali z veliko manjkajočimi vrednostmi ciljne spremenljivke (Blockeel, 1998). Do sedaj je bilo napovedno razvrščanje omejeno na metode za gradnjo odločitvenih dreves (Blockeel, 1998; Blockeel et al., 1998).

A.1.4 Učenje pravil

Množice odločitvenih pravil »če-potem« sodijo med človeku najbolj razumljive oblike napovednih modelov (Flach in Lavrač, 2003; Mitchell, 1997). V primerjavi z odločitvenimi drevesi, kjer moramo celotno drevo interpretirati kot celoto, lahko posamezna pravila interpretiramo vsako posebej. Pravila znotraj množice so oblike *'ČE pogoj POTEM napoved'*, poleg tega pa imamo v množici pravil običajno tudi *privzeto pravilo* s katerim klasificiramo primere, ki ne ustrezajo pogojem nobenega drugega pravila. Za primere, ki ustrezajo pogoju nekega pravila pravimo, da jih to pravilo *pokriva.* Ločimo *urejene* in *neurejene* množice pravil. V urejenih množicah pravil, ali *odločitvenih seznamih,* so pravila urejena v seznam in prvo pravilo v seznamu, ki pokrije dani primer, uporabimo za napoved razreda primera. Pri neurejenih množicah pravil pa zberemo napovedi vseh pravil, ki pokrijejo primer ter jih sestavimo v končno napoved razreda primera.

Večina metod za gradnjo klasifikacijskih pravil temelji na prekrivnem algoritmu (Michalski, 1969; Michalski et al., 1986). Med njimi je zelo znana metoda CN2 (Clark in Niblett, 1989; Clark in Boswell, 1991). Metoda CN2 iterativno gradi pravila, ki pokrivajo primere s podobnimi vrednostmi ciljne spremenljivke. Za preiskovanje prostora vseh možnih pravil uporablja hevristično iskanje, pri čemer je hevristika kar natančnost grajenih pravil. Zgrajeno pravilo dodamo v množico pravil, hkrati pa iz učne množice odstranimo vse primere, ki jih je to pravilo pokrilo. Nato postopek ponavljamo, dokler v učni množici ne zmanjka primerov ali ne moremo več najti nobenega pravila. Pravila zgrajena na ta način so urejena. Z metodo CN2 lahko gradimo tudi neurejena pravila, če pri gradnji pravil iz učne množice vsakič odstranimo le tiste primere, ki jih novo pravilo pravilno klasificira in če postopek gradnje ponovimo za vsako možno vrednost ciljne spremenljivke. Večina metod za gradnjo pravil je namenjena gradnji klasifikacijskih pravil, obstaja pa tudi nekaj metod za regresijska pravila. Sistem R^2 (Torgo, 1995) uporablja nekoliko spremenjen prekrivni algoritem, model v vsakem pravilu je lahko linearna funkcija ali pa le konstantna vrednost. Zgrajena pravila so neurejena. FORS (Karalič in Bratko, 1997) je sistem za induktivno logično programiranje in kot tak za opis učnih primerov uporablja logiko prvega reda. Gradi urejena pravila in pri tem, tako kot CN2, uporablja prekrivni algoritem.

Z gradnjo pravil je povezano tudi področje odkrivanja podskupin (Lavrač et al., 2004), ki se ukvarja z iskanjem in opisovanjem zanimivih (pod)skupin primerov. Same metode za odkrivanje podskupin so zelo podobne metodam za gradnjo pravil, vendar pa uporabljajo nekatere zanimive rešitve, kot je na primer *uteženi prekrivni algoritem*, ki je predstavljal izhodišče za razvoj bolj splošnega prekrivnega algoritma, ki deluje na klasifikacijskih in regresijskih problemih z eno ali večimi ciljnimi spremenljivkami.

A.2 Pregled vsebine

Disertacija je sestavljena iz šestih poglavij. Prvo poglavje podaja uvod v disertacijo s poudarkom na zastavljenih ciljih in poglavitnih prispevkih znanosti. Drugo poglavje poda ozadje napovednega modeliranja, razvrščanja v skupine, napovednega razvrščanja in učenja pravil. Tretje poglavje opisuje mere za ocenjevanje kvalitete posameznih pravil in množic pravil. Te mere so uporabljene v samem učnem algoritmu za pravila za napovedno razvrščanje, ki je opisan v naslednjem, četrtem poglavju. To poglavje predstavlja glavni del disertacije. Peto poglavje predstavlja empirično evalvacijo metode za učenje pravil za napovedno razvrščanje. Zadnje poglavje podaja sklep disertacije in na kratko izpostavi izvirne prispevke disertacije ter rezultate evalvacije novo razvitih metod. Poglavje se konča s smernicami za nadaljnje delo. V nadaljevanju tega razdelka bomo predstavili nekaj bistvenih prispevkov disertacije.

A.2.1 Ocenjevanje kvalitete pravil in množic pravil

Preden lahko začnemo z učenjem posameznih pravil in množic pravil, moramo določiti zahteve, katerim mora zadoščati vsako posamezno pravilo ter množica pravil kot celota. V praksi si te zahteve med seboj pogosto nasprotujejo, in si moramo bodisi izbrati eno najpomembnejšo zahtevo, ali pa med večimi poiskati primeren kompromis. Od množice pravil tako na primer pogosto zahtevamo, da je majhna, in da ima majhno napovedno napako, vendar imajo lahko zelo majhne množice pravil veliko napovedno napako. V nadaljevanju bomo našteli nekaj mer za ocenjevanje kvalitete pravil in množic pravil s katerimi si pomagamo pri učenju in ocenjevanju ze naučenih množic pravil.

Posamezna pravila

Za vsako posamezno pravilo želimo, da nam podaja neko novo in splošno informacijo o obravnavani problemski domeni. Zato mora biti pravilo najprej točno oz. mora imeti *majhno napovedno napako.* Nadalje, če naj bo pravilo splošno, mora *pokrivati veliko število učnih primerov.* Vsako pravilo naj bi nam podajalo neko novo informacijo o obravnavani domeni, zato naj ne bi pokrivalo primerov, ki jih že pokrivajo ostala pravila iz množice; želimo torej *veliko razdaljo med novim pravilom in ze obstoječimi pravili.* Poleg tega naj bi vsako pravilo *pokrivalo skupino primerov katere statistični opis (prototip) se razlikuje od statističnega opisa celotne učne množice.* Vsako od omenjenih zahtev lahko bolj natančno podamo z definicijo ustrezne mere. V strojnem učenju v ta namen (npr. za napovedno napako) običajno uporabljamo mere, ki so primerne le za eno samo nominalno ali numerično spremenljivko. Za potrebe napovednega razvrščanja pa potrebujemo mere, ki so uporabne tudi na domenah z več nominalnimi ali numeričnimi ciljnimi spremenljivkami (atributi).

Namesto napovedne napake lahko npr. definiramo bolj splošno mero, poimenovali smo jo *disperzija,* ki je uporabna tudi za ocenjevanje točnosti (oziroma bolje rečeno kompaktnosti primerov, ki jih pravilo pokrije) pravil za napovedno razvrščanje z več ciljnimi atributi. Disperzijo prek več atributov zapišemo kot uteženo povprečje disperzij po posameznih atributih. Uteži nam na primer omogočajo, da damo večji poudarek atributom, za katere na podlagi domenskega predznanja sklepamo, da so pomembnejši od ostalih. Disperzijo vzdolž enega nominalnega atributa definiramo kot (normirano) povprečno razdaljo enega primera iz množice do prototipa te množice. Prototip ali prototipni primer množice primerov vzdolž nominalnega atributa je enak vektorju relativnih frekvenc vrednosti, ki jih ta atribut lahko zavzame. Po enakem principu bi lahko definirali tudi disperzijo za numerične atribute, vendar smo se raje odločili, da disperzijo numeričnih atributov merimo z varianco. Varianca je namreč v statistiki in strojnem učenju pogosto uporabljana mera z znanimi lastnostmi, ki ustreza tudi našim potrebam.

Množice pravil

V strojnem učenju se uporablja veliko različnih mer za ocenjevanje množic pravil in napovednih modelov v splošnem. Za klasifikacijske naloge je daleč najbolj pogosta *klasifikacijska točnost*, oziroma *klasifikacijska napaka*. Nekatere druge pogosto uporabljane mere, kot na primer *preciznost*, *priklic*, in *ploščina pod krivuljo ROC* niso primerne za klasifikacijske probleme z več kot dvema možnima razredoma, kaj šele za klasifikacijske probleme z več ciljnimi spremenljivkami. Za regresijske naloge sta najbolj pogosto uporabljani meri *relativni koren srednje kvadratne napake (angl. relative root mean squared error (RRMSE))* in *korelacijski koeficient*, pogosto pa se uporabljajo tudi *koren srednje kvadratne napake (angl. root mean squared error (RRMSE))*, *srednja kvadratna napaka (angl. mean squared error (MSE))*, in *srednja absolutna napaka (angl. mean absolute error (MAE))*.

A.2.2 Učenje pravil za napovedno razvrščanje

Problem gradnje napovednih pravil definiramo na naslednji način. Podane imamo: opisni atributni prostor $\mathcal{D}$, ciljni atributni prostor $\mathcal{T}$, množico N primerov $\{\mathbf{e}_i\}_1^N$, kjer je $\mathbf{e}_i = [\mathbf{x}_i, \mathbf{y}_i] \in \mathcal{D} \times \mathcal{T}$, opisni jezik B nad $\mathcal{D}$, mero razdalje med dvema primeroma $d(\mathbf{e}_i, \mathbf{e}_j)$ in prototipno funkcijo $p(\{\mathbf{e}_k\}_1^K)$, ki množici primerov pripiše prototipni primer. Iščemo množico skupin primerov, za katere velja naslednje. Vsaki skupini pripada opis v jeziku B in vsaki skupini pripada napoved izražena kot prototip te skupine. Poleg tega so razdalje med primeri znotraj skupin majhne in razdalje med primeri v različnih skupinah so velike. Vsaka skupina naj bo predstavljena s pravilom oblike *'ČE <opis skupine> POTEM <prototip skupine>'*. Takšna definicija problema hkrati zaobjema nalogo razvrščanja (prostor $\mathcal{T} = \emptyset$), nalogi klasifikacije in regresije (mera razdalje d upošteva le projekcijo primerov na prostor $\mathcal{T}$) ter nalogo napovedovanja več (nominalnih ali numeričnih) ciljnih spremenljivk hkrati (prostor $\mathcal{T}$ je večdimenzionalen).

Učni algoritem

Kot smo že omenili, velika večina metod za učenje pravil temelji na *prekrivnem algoritmu* (Michalski, 1969; Michalski et al., 1986). Za učenje pravil za napovedno razvrščanje smo razvili bolj splošen algoritem, ki zaobjema tudi prekrivni algoritem. Algoritem za učenje pravil za napovedno razvrščanje, oziroma njegov zgornji nivo, je predstavljen v tabeli A.1. Učenje začnemo s prazno množico pravil R in z množico učnih primerov E. V vsaki iteraciji poiščemo množico kandidatnih pravil R_c. Kandidatna pravila nato ovrednotimo ter najboljše med njimi (r_i) (če obstaja in dosega podane minimalne kriterije) dodamo v množico pravil R. Nato z namenom, da bomo v naslednji iteraciji našli drugačna kandidatna pravila, spremenimo trenutno učno množico primerov E_c. Zanko ponavljamo, dokler ni izpolnjen ustavitveni pogoj

Tabela A.1: Zgornji nivo algoritma za učenje pravil za napovedno razvrščanje.

E	... začetna učna množica
E_c	... učna množica v trenutni iteraciji
R	... množica pravil
R_c	... množica kandidatnih pravil
r_i	... pravilo dodano v trenutni iteraciji

```
procedure NaučiMnožicoPravil($E$)
    $R = \emptyset$
    $E_c = E$
    repeat
        $R_c = \text{PoiščiKandidatnaPravila}(E_c)$
        $r_i = \text{NajboljšePravilo}(R_c, R)$
        if ($r_i \neq \emptyset$) then
            $R = R \cup \{r_i\}$
        $E_c = \text{SpremeniUčnoMnožico}(E_c, r_i)$
    until KončajUčenje($E_c, R, r_i$)
    $R = R \cup \text{PrivzetoPravilo}(E)$
    $R = \text{OptimirajMnožicoPravil}(R, E)$
    return $R$
```

('KončajUčenje'). Nato dodamo še privzeto pravilo, to je pravilo za klasificiranje primerov, ki jih ne pokrije nobeno drugo pravilo iz množice. Preden končamo z učenjem množice pravil, lahko naučena pravila še optimiramo, na primer tako, da pravila v množici poenostavimo.

Hevristično preiskovanje prostora vseh možnih pravil je daleč najbolj pogosto uporabljena metoda učenja pravil znotraj algoritmov za učenje pravil. Algoritem za učenje pravil za napovedno razvrščanje uporablja zelo podoben algoritem kot metoda CN2 (Clark in Niblett, 1989; Clark in Boswell, 1991). Ključno vlogo pri tem algoritmu igra hevristična funkcija, katere namen je ocenjevanje različnih pravil in nam služi za vodenje iskanja v smeri pravil željene kvalitete. Hevristična funkcija naj bi merila kvaliteto vsakega pravila posebej in v kombinaciji s celotno množico pravil. V prvem delu razdelka A.2.1 smo govorili o zahtevanih lastnostih pravil, kot tudi o merah za njihovo ocenjevanje. V najbolj splošnem primeru lahko v hevristično funkcijo vključimo vse štiri omenjene mere. Seveda je možnih načinov za kombinacijo mer več, mi si bomo za vzor vzeli hevristiko WRAcc (Lavrač et al., 1999) ter omenjene mere med sebo pomnožili. Naj bo torej c pogoj pravila r, ki ga ocenjujemo, in naj bo E množica vseh učnih primerov. E_r je podmnožica primerov, ki izpolnjujejo pogoj c (primeri, ki jih pokriva pravilo r) in R je množica do sedaj zgrajenih pravil. Hevristično funkcijo sedaj zapišemo kot (večja vrednost funkcije pomeni boljše pravilo)

$$h^*(c) = [d_{\mathit{off}} - \mathrm{disp}(E_r; \mathbf{w}_a)] \cdot \mathrm{cov}(r; E, \mathbf{w}_e)^{\alpha} \cdot \mathrm{dist}(r, R)^{\beta} \cdot \mathrm{diss}(r; E, \mathbf{w}_a)^{\gamma}. \tag{A.1}$$

Pri tem disperzija $\mathrm{disp}(E_r; \mathbf{w}_a)$ meri točnost pravila, $\mathrm{cov}(r; E, \mathbf{w}_e)$ pomeni število primerov (oz. vsoto njihovih uteži), ki jih pokriva pravilo, $\mathrm{dist}(r, R)$ pomeni oddaljenost pravila od ostalih pravil v množici R in $\mathrm{diss}(r; E, \mathbf{w}_a)$ pomeni različnost prototipa pravila od prototipa celotne učne množice. Parametri α, β in γ nam omogočajo, da bolj poudarimo eno ali drugo mero. Parameter d_{off} smo uvedli po vzoru hevristike WRAcc (Lavrač et al., 1999); če ga nastavimo na vrednost disperzije celotne učne množice E, lahko na prvi faktor enačbe gledamo kot na relativno zmanjšanje disperzije. Če hočemo v celoti posnemati hevristiko WRAcc, moramo dodatno nastaviti še $\alpha = 1$ in $\beta, \gamma = 0$. Poleg omenjenih parametrov v hevristični funkciji nastopajo še uteži atributov $\mathbf{w}_a$ in uteži učnih primerov $\mathbf{w}_e$. Slednje pridejo v poštev v kombinaciji z uteženim prekrivnim algoritmom, ki ga bomo opisali v nadaljevanju. Uteži atributov nam načeloma omogočajo, da določamo vpliv vsakega atributa posebej. V praksi pa atribute vedno razdelimo na opisne in ciljne ter njihove uteži določimo takole:

$$w_{aj} = w_{a_j} = \begin{cases} \tau, & a_j \text{ je ciljni atribut,} \\ 1 - \tau, & \text{sicer,} \end{cases} \tag{A.2}$$

pri čemer naj parameter τ zadošča pogoju $0 < \tau \leq 1$. Če je naš glavni cilj učenje čim bolj točnih pravil, potem nastavimo $\tau = 1$. Po drugi strani pa z nastavitvijo τ na vrednost manjšo od 1 zahtevamo, da vsako pravilo pokriva primere, ki so si čim bolj podobni po vrednostih vseh atributov. Parameter τ nam pravzaprav omogoča prehajanje med napovednim modeliranjem na eni strani in razvrščanjem v skupine na drugi strani.

Standardni prekrivni algoritem spremeni množico E_c (procedura 'SpremeniUčnoMnožico') tako, da iz nje odstrani primere, ki jih je pokrilo pravilo dodano v tej iteraciji. Na ta način pri iskanju novih pravil (procedura 'PoiščiKandidatnaPravila') bolj poudarimo primere, ki še niso bili primerno pokriti. Ta pristop uporablja tudi algoritem CN2 (Clark in Niblett, 1989; Clark in Boswell, 1991). Nekoliko drugače spremeni trenutno učno množico primerov *uteženi prekrivni algoritem* (Gamberger in Lavrač, 2002). Le-ta vsakemu učnemu primeru priredi utež, ki je na začetku enaka ena. Namesto, da bi novo pokrite primere popolnoma odstranil iz učne množice, jim le zmanjša utež. Vendar uteži ne zmanjša vsem novo pokritim primerom, pač pa le tistim, ki jih je novo dodano pravilo pravilno klasificiralo. Žal je pojem »pravilno klasificiranega primera« definiran le za klasifikacijske probleme z eno ciljno spremenljivko. Zato smo razvili bolj splošno prekrivno metodo, poimenovali smo jo *z napako uteženi prekrivni algoritem,* ki deluje na klasifikacijskih in regresijskih domenah z eno ali več ciljnimi spremenljivkami. Z napako uteženi prekrivni algoritem je podoben »običajnemu« uteženemu prekrivnemu algoritmu, le da je vrednost za katero zmanjšamo utež danega učnega primera sorazmerna z napako, ki jo novo dodano pravilo naredi pri napovedovanju vrednosti ciljnih spremenljivk tega primera.

A.2.3 Eksperimentalna evalvacija

Učinkovitost novo razvitih metod za učenje pravil za napovedno razvrščanje smo empirično ovrednotili z več skupinami poskusov. Predvsem nas je zanimala napovedna točnost (klasifikacijska napaka za klasifikacijske ter relativni koren srednje kvadratne napake in korelacijski koeficient za regresijske probleme) ter velikost naučenih množic pravil. Vse ocene napak so bile narejene s postopkom 10-kratnega prečnega preverjanja. Signifikantnost opaženih razlik med posameznimi metodami smo preverili z uporabo Wilcoxonovega testa rangiranih predznakov (angl. signed-rank test) (Wilcoxon, 1945). Najprej smo nove metode preizkusili na klasifikacijskih in regresijskih problemih z eno ciljno spremenljivko ter jih primerjali z nekaterimi obstoječimi metodami (CN2, CN2-WRAcc, CN2-EVC, JRip in FRS); pri tem smo uporabili izbor standardnih testnih podatkov (Newman et al., 1998). Učinkovitost napovedovanja več ciljnih spremenljivk hkrati smo ovrednotili s primerjavo z modeli za napovedovanje ene same ciljne spremenljivke ter z modeli, zgrajenimi z drevesi za napovedno razvrščanje; pri tem smo uporabili več realnih množic podatkov.

Primerjava z obstoječimi metodami

Poskusi na klasifikacijskih problemih z eno ciljno spremenljivko kažejo, da se točnost urejenih pravil za napovedno razvrščanje (PNR) signifikantno ne razlikuje od urejenih pravil naučenih z metodama CN2 in CN2-WRAcc. Urejena PNR se tudi ne razlikujejo signifikantno od dreves za napovedno razvrščanje (DNR), čeprav so nekoliko manj točna. Urejena pravila JRip so boljša od urejenih PNR (*p-vrednost*=0.07). Neurejena PNR so boljša kot neurejena pravila metod CN2 in CN2-WRAcc zaradi uporabe z napako uteženega prekrivnega algoritma za učenje neurejenih PNR. Točnost neurejenih PNR se signifikantno ne razlikuje od točnosti neurejenih pravil CN2-EVC, urejenih pravil JRip in dreves (DNR), čeprav so omenjene metode nekoliko manj točne.

Rezultati evalvacije na regresijskih problemih z eno ciljno spremenljivko kažejo, da so tako urejena kot neurejena PNR bolj točna od (urejenih) pravil FRS. Po drugi strani pa primerjava z drevesi (DNR) kaže, da so PNR signifikantno slabša od dreves.

Evalvacija na klasifikacijskih problemih z več spremenljivkami kaže, da so urejena PNR nekoliko slabša od dreves (DNR), vendar razlika ni signifikantna. Poleg tega so množice PNR običajno manjše od (v pravila prepisanih) dreves. Podobno velja tudi za neurejena PNR, le da so tokrat PNR nekoliko boljša od dreves.

Poskusi na regresijskih problemih z več ciljnimi spremenljivkami in primerjava PNR z drevesi nam kažejo podobno sliko kot v primeru regresije ene ciljne spremenljivke. Tako urejena kot neurejena PNR so signifikantno slabša od dreves (DNR). Domnevamo, da je glavni razlog za omenjeno razliko pri regresijskih problemih v tem, da DNR uporabljajo zelo učinkovito metodo naknadnega rezanja regresijskih dreves, medtem ko algoritem PNR ne vsebuje nobene metode poenostavljanja in izločanja nepotrebnih pravil.

Primerjava med napovedovanjem ene in večih ciljnih spremenljivk

Primerjava PNR za napovedovanje več ciljnih spremenljivk z ustreznimi množicami PNR za napovedovanje ene ciljne spremenljivke na klasifikacijskih problemih kaže, da so v primeru urejenih pravil boljša pravila za napovedovanje ene ciljne spremenljivke, medtem ko so v primeru neurejenih pravil boljša pravila za napovedovanje več spremenljivk. Razlike v obeh primerih niso signifikantne (*p-vrednost*=0.07). Po drugi strani pa so razlike v velikosti množic pravil zelo signifikantne. Urejene in neurejene množice pravil za napovedovanje več ciljnih spremenljivk so veliko manjše od ustreznih zbirk množic pravil za napovedovanje ene ciljne spremenljivke.

Rezultati evalvacije na regresijskih problemih z več ciljnimi spremenljivkami si med seboj nekoliko nasprotujejo, vendar na njihovi osnovi ne moremo sklepati, da sta točnost PNR za napovedovanje več ciljnih spremenljivk in točnost PNR za napovedovanje ene ciljne spremenljivke signifikantno različni. Po drugi strani pa je velikost množic pravil za napovedovanje več ciljnih spremenljivk ponovno signifikantno manjša.

A.3 Izvirni prispevki disertacije

Pričujoča doktorska disertacija vsebuje več izvirnih prispevkov k področju strojnega učenja. Tako smo razvili novo metodo za učenje neurejenih pravil za klasifikacijo ene ciljne spremenljivke. Metoda temelji na znani metodi CN2 (Clark in Niblett, 1989; Clark in Boswell, 1991), vendar uporablja posplošeni *uteženi prekrivni algoritem* (Gamberger in Lavrač, 2002).

Omenjeno metodo smo nadalje posplošili tako, da omogoča učenje urejenih ali neurejenih pravil, na klasifikacijskih ali regresijskih problemih z eno ali več ciljnimi spremenljivkami. Metoda uporablja iskalno hevristiko, ki upošteva več mer za kvaliteto pravil hkrati in je uporabna na vseh zgoraj omenjenih tipih problemskih domen.

Tretji prispevek disertacije je razširitev pristopa napovednega razvrščanja na modele predstavljene v obliki pravil. Novo razvita metoda enotno obravnava učenje pravil in razvrščanje v skupine. Uporabljena iskalna hevristika upošteva tako ciljne kot tudi opisne spremenljivke. Različna obtežitev obeh skupin spremenljivk nam omogoča prehajanje med napovednim modeliranjem in razvrščanjem v skupine. Večja obtežitev ciljnih spremenljivk nam zagotavlja večjo napovedno točnost naučenih pravil, medtem ko ob večji obtežitvi opisnih spremenljivk dobimo pravila, ki opisujejo bolj kompaktne skupine primerov. Pričakujemo, da bo razširitev napovednega razvrščanja na modele zapisane v obliki pravil omogočila njegovo uporabo na novih problemskih domenah, kjer so poleg napovedne točnosti pomembne tudi druge lastnosti zgrajenih modelov, kot je na primer njihova razumljivost.

Sklepni prispevek disertacije je obširna empirična evalvacija novo razvite metode na klasifikacijskih in regresijskih problemih z eno in več ciljnimi spremenljivkami. Metodo smo primerjali z obstoječimi metodami. Rezultati primerjave na klasifikacijskih problemih z eno ciljno spremenljivko kažejo, da je točnost pravil za napovedno razvrščanje (PNR) primerljiva s točnostjo pravil, zgrajenih z metodo CN2 ter dreves za napovedno razvrščanje (DNR), medtem ko so neurejena PNR boljša od neurejenih CN2 pravil. Neurejena PNR so v splošnem boljša od urejenih PNR. Na klasifikacijskih problemih z več ciljnimi spremenljivkami so PNR primerljiva z DNR, vendar z metodo PNR običajno dobimo manj pravil, kot če DNR prepišemo v pravila. Regresijska PNR za napovedovanje ene ciljne spremenljivke so primerljiva s pravili FRS, vendar pa so precej slabša kot DNR; enako velja za regresijske probleme z več ciljnimi spremenljivkami, kjer so PNR tudi precej slabša od DNR. Primerjava točnosti PNR za napovedovanje ene ciljne spremenljivke ter PNR za napovedovanje več ciljnih spremenljivk hkrati pokaže, da nam oba tipa modelov omogočata primerljivo napovedno točnost, vendar pa so množice PNR za napovedovanje več ciljnih spremenljivk hkrati bistveno manjše od ustreznih množic PNR za napovedovanje le ene ciljne spremenljivke.

V pričujoči disertaciji smo predstavili precej splošen algoritem za učenje pravil, ki zaobjema več različnih metod za učenje pravil. Raziskali in empirično evalvirali smo le osnovni pristop k učenju pravil za napovedno razvrščanje, ki temelji na prekrivnem algoritmu. V disertaciji smo poleg tega predstavili več idej in odprtih problemov, ki odpirajo obilo možnosti za nadaljnje raziskave.

Izjava o avtorstvu

Izjavljam, da sem doktorsko disertacijo izdelal samostojno na Odseku za tehnologije znanja Instituta Jožef Stefan pod vodstvom somentorja prof. dr. Saša Džeroskega ter s pomočjo mentorja akad. prof. dr. Ivana Bratka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Mag. Bernard Ženko