# Learning Classification Rules for Multiple Target Attributes

Bernard Ženko and Sašo Džeroski

Department of Knowledge Technologies, Jožef Stefan Institute
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
`Bernard.Zenko@ijs.si, Saso.Dzeroski@ijs.si`

**Abstract.** Among predictive models, 'if-then' rule sets are one of the most expressive and human readable model representations. Most of the existing approaches for rule learning focus on predicting a single target attribute/class. In practice, however, we encounter many problems where the task is to predict not one, but several related target attributes. We employ the predictive clustering approach to learn rules for simultaneous prediction of multiple target attributes. We propose a new rule learning algorithm, which (unlike existing rule learning approaches) generalizes to multiple target prediction. We empirically evaluate the new method and show that rule sets for multiple target prediction yield comparable accuracy to the respective collection of single target rule sets. The size of the multiple target rule set, however, is much smaller than the total size of the collection of single target rule sets.

## 1 Introduction

Traditionally, inductive machine learning focuses on problems where the task is to predict a value of a single target attribute. However, there exist many real life problems where the task is to predict not one, but several related target attributes. Of course, this problem can easily be solved by constructing separate models for each target attribute. If our only goal is to achieve high predictive accuracy, the resulting collection of (single target) models should be sufficient, provided that we have selected a suitable method for single target prediction. On the other hand, if, besides the predictive accuracy, the interpretability of induced models is also important, the collection of single target models is far less understandable than a single model that jointly predicts all target attributes. Therefore, the research on extending machine learning methods that produce interpretable models (such as decision trees and rules) towards multiple target prediction is justified.

One of the possible approaches to multiple target prediction is predictive clustering, which was originally applied to decision trees. The goal of this paper is to adopt the predictive clustering approach in order to design a method for learning rules for multiple target prediction; we call it *predictive clustering rules*.[1] We focus on classification tasks only, though the method can also be extended to regression problems.

---

[1] An initial solution to the problem of learning predictive clustering rules has been presented in [17]. The algorithm presented here includes an updated search heuristic, a new error weighted covering algorithm, and extended experimental evaluation.

The rest of the paper is organized as follows. Section 2 summarizes predictive clustering. The algorithm for learning predictive clustering rules is presented in Section 3. Section 4 describes the evaluation methodology, and Section 5 presents the experimental results. Last section concludes and gives some directions for further work.

## 2   Predictive Clustering

The predictive clustering approach [1,2] builds on ideas from two machine learning areas, predictive modeling and clustering [9]. 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 (attribute-value representation is most commonly used for describing objects and their properties). 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, (though a symbolic descriptions can be added to already constructed clusters as in *conceptual clustering* [13]). Both areas are usually regarded as completely different tasks. However, predictive modeling methods that partition the example space, such as decision trees and rules are also very similar to clustering [10]. 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 builds on this similarity. 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 collection of models, each predicting a single variable.[2] So far, this approach has been limited to the tree learning methods. The method described in the next section extends predictive clustering towards methods for learning rules.

## 3   Predictive Clustering Rules

Predictive clustering rules (PCRs) include ideas from rule learning and clustering. The learning algorithm itself is a generalization of existing rule learning approaches. The rule evaluation function which serves as a search heuristic, however, employs techniques commonly used in clustering. We start with a description of the top level of the algorithm, while specific aspects of the algorithm, such as learning single rules and modification of the learning set between subsequent iterations of the algorithm, are discussed in separate sections.

---

[2] Related to multiple target prediction is *Multi-Task learning* [3], where a single model (neural network) is trained for multiple target attributes (learning tasks) with a presumption that a set of hidden submodels will emerge that are used for modeling of all learning tasks.

**Table 1.** The algorithm for learning predictive clustering rules. **a)** 'LearnRuleSet', **b)** 'FindCandidateRule', and **c)** 'ModifyLearningSet' procedures.

---

**a) LearnRuleSet()**

  **Input:** learning set $E$
  $R = \emptyset$ {rule set}
  $E_c = E$ {current learning set}
  **repeat**
    $r_i = \text{FindCandidateRule}(E_c)$
    $R = R \cup \{r_i\}$
    $E_c = \text{ModifyLearningSet}(E_c, r_i)$
  **until** $((r_i = \emptyset)$ **or** $(\|E_c\| = 0))$
  $R = R \cup \text{DefaultRule}(E)$
  **return** $R$

**c) ModifyLearningSet()**

  **Input:** current learning set $E_c$, newly added
  rule $r_i$
  **case** $(M_{mod} = \text{“Std-Covering”})$ **do**
    **for all** $(\mathbf{e}_i \in E_c)$ **do**
      **if** $(r_i$ covers $\mathbf{e}_i)$ **then**
        $w_{ei} = 0$
    **return** $E_c$
  **case** $(M_{mod} = \text{“Err-Weight-Covering”})$ **do**
    **for all** $(\mathbf{e}_i \in E_c)$ **do**
      **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** $E_c$

**b) FindCandidateRule()**

  **Input:** current learning set $E_c$
  $c_{last} = \text{“true”}$
  $C = C_{best} = \{c_{last}\}$
  **while** $(C \neq \emptyset)$ **do**
    $C_{new} = \emptyset$
    **for all** $(c \in C)$ **do**
      **for all** $(t \in T_p \wedge t \notin c)$ **do**
        $c_{new} = c \wedge t$
        **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** $(|C_{best}| > b_w)$ **then**
          $C_{best} = C_{best} \setminus \arg\min_{c' \in C_{best}} h(c')$
        $c_{last} = \arg\min_{c' \in C_{best}} h(c')$
    $C = C_{new}$
  $c_{best} = \arg\max_{c' \in C_{best}} h(c')$
  **return** $(c_{best}, p_{best})$

---

## 3.1 Learning Algorithm

Most of existing approaches to rule learning are based on the *covering algorithm* [12]. Its main problem, however, is that it was originally designed for two-class (binary) classification problem domains. In addition, the rule sets produced by the original covering algorithm are by nature ordered, unless rules for only one class value are constructed. Our algorithm is based on the CN2 method [5,4], which uses a version of the covering algorithm that can learn ordered or unordered rules, and is also applicable to (single target) multi-class problems.

The algorithm for learning predictive clustering rules is presented in Table 1. Top level procedure 'LearnRuleSet' (Table 1.a) starts with an empty rule set $R$ and a set of learning examples $E$. In each iteration we learn a candidate rule $r_i$ and add it to the rule set. Next, we modify the current learning set $E_c$ and, unless some stopping criterion is met, repeat the loop. There are two stopping criteria; we stop adding rules if the 'FindCandidateRule' procedure could not find any non-empty rule, and when the $\|E_c\|$ becomes zero ($\|E_c\|$ is the number of examples with non-zero weights). Before the

learning procedure is finished, we add the default rule. The default rule is a rule with an empty condition and is used for examples that are not covered by any other rule. Its prediction part is a cluster prototype of the complete learning set $E$.

The interpretation of PCRs is the same as that of CN2 rules: ordered rules are scanned and the first one that covers the example is used; predictions of all unordered rules that cover the example are combined into the final prediction via weighted voting, where the weights are equal to the number of covered examples on the training data.

## 3.2   Learning Single Rule

The 'FindCandidateRule' procedure is given in Table 1.b, and is a general-to-specific beam search algorithm, which is very similar to the one implemented in the CN2. The input to the procedure is the learning set of examples $E_c$. The width of the beam $b_w$ determines the number of partial rules maintained during the search. A set of $b_w$ 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 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. Here we only consider conditions that cover at least a predefined minimal number of examples $\mu$. 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 best condition from the set $C_{best}$ is coupled with the prototype of target attributes of examples that it covers $(p_{best})$, and returned as a candidate rule.

**Search Heuristic.**   The crucial part of the algorithm is the search heuristic $h$. The heuristic is used for the evaluation of rules under construction and basically leads the search procedure towards rules of the desired quality. Therefore, the heuristic should reflect the qualities we expect from each individual rule in the rule set. Typically, we want the rules to be accurate and, at the same time, general, i.e., we want the rules to cover as many examples as possible. 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.

Normally, the accuracy measures are tailored to single target prediction, while for predictive clustering rules we need a measure that also works for multiple target prediction. We define such a measure, we call it *dispersion,* as follows. Let $E'$ be the set of $N$ examples that are covered by a specific rule, and each example $\mathbf{e}_i$ is represented as a vector of $K$ attribute values $x_{ji}$, where $x_{ji}$ stands for the value of the attribute $a_j$ of the example $\mathbf{e}_i$. The dispersion of a set of examples $E'$ is an average of the dispersions along each attribute

$$\text{disp}(E') = \frac{1}{K} \sum_{j=1}^{K} \text{disp}(E', a_j). \tag{1}$$

Here we take into account only the target attributes, although in principle, we could include also the non-target attributes [17].

The definition of dispersion along a single nominal attribute 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 $f_k$ of possible values within the set

$$\mathbf{p}_{E'} = \mathbf{p}(E'; a_j) = [f_1, f_2, \ldots, f_L]; \quad f_k = \frac{n_k}{N}, \tag{2}$$

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

$$\mathbf{p}_{\mathbf{e}_i} = [f_1', f_2', \ldots, f_L']; \quad f_k' = \begin{cases} 1, & \text{if } x_{ji} = l_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

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

$$d(\mathbf{p}_{\mathbf{e}_i}, \mathbf{p}_{E'}) = |1 - f_k| + \sum_{\substack{m=1 \\ m \neq k}}^{L} |f_m| = 2(1 - f_k); \tag{4}$$

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

$$\text{disp}(E', a_j) = \frac{1}{2N} \frac{L}{L-1} \sum_{i=1}^{N} d(\mathbf{p}_{\mathbf{e}_i}, \mathbf{p}_{E'}). \tag{5}$$

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.

The rule's generality is typically measured by its *coverage,* which is defined as the proportion of covered examples, i.e., the number of examples covered by a rule divided by the number of all examples. This definition assumes that all examples are equally important, i.e., they all have equal weight. As we will see later, sometimes it is useful to introduce example weights that are not uniform. Each example $\mathbf{e}_i$ then has an associated weight $w_{ei}$. 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

$$\text{cov}(r; E, \mathbf{w}) = \frac{\sum_{\mathbf{e}_i \in E'} w_i}{\sum_{\mathbf{e}_i \in E} w_{ei}}. \tag{6}$$

Now, we have to combine the two measures into a single heuristic function. Analogously to the WRAcc heuristic [11], we do this as follows. Let $c$ be the condition 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 $r$). $\mathbf{w}_e$ is the example

weight vector. 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 (more on this later). The heuristic function is

$$h^*(c) = [d_{def} - \text{disp}(E_r)] \cdot \text{cov}(r; E, \mathbf{w}_e)^\alpha. \tag{7}$$

The parameter $\alpha$ enables us to put more (or less) emphasis on coverage w.r.t. to dispersion; by default (like in WRAcc) it is set to 1. $d_{def}$ is the *default dispersion,* i.e., the dispersion of the entire learning set $E$, and the first factor of Equation 7 can be regarded as the relative dispersion loss. Rules with larger heuristic function values are better.

### 3.3   Modifying the Learning Set

Within the main loop of the 'LearnRuleSet', the current learning set $E_c$ must be modified, otherwise the 'FindCandidateRule' procedure would continuously find the same rule. Learning set modification is done by the 'ModifyLearningSet' procedure presented in Table 1.c.

The most common approach to modifying the learning set is the covering algorithm [12]. 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 [5,4] for the induction of ordered rules, and ordered PCRs are also induced in this manner.

The weighted covering algorithm [8], 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 prediction 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 as

$$g(\mathbf{e}_i, r) = 1 + (\zeta - 1)k(\mathbf{e}_i, r), \tag{8}$$

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

$$k(\mathbf{e}_i, r) = \frac{\text{nb. corr. pred. tar. atts of } \mathbf{e}_i \text{ by } r}{\text{nb. all tar. atts}}, \tag{9}$$

and $\zeta$ is the *covering weight parameter,* which enables us, together with the *covering weight threshold parameter $\epsilon$,* to control the pace of removing covered examples from the current learning set. It should take values between 0 and 1. Setting $\zeta$ to 0 means that examples, whose target attributes are correctly predicted by rule $r$, are immediately removed from the current learning set, i.e., their weights are set to zero. The parameter $\epsilon$ 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.

## 4   Experimental Setup

In the experiments, we investigate two issues. First, we compare the performance of predictive clustering rules (PCRs) to some existing rule learning methods for single target classification in order to show that PCRs are comparable to existing methods on this type of problems, and can be used as a baseline in the second part of the evaluation. For comparison we selected the CN2 rule learner [5,4] and a modification of CN2, the CN2-WRAcc [15], because our approach is a generalization of these algorithms. Additionally, we compare PCRs to Ripper [6] which is a more advanced rule learner; we used the JRip implementation from the Weka data mining suite [16] which only learns ordered rules.

Second, we compare the PCRs for single target prediction to PCRs for multiple target prediction. The main benefit of multiple target prediction is that a collection of models (rule sets) each predicting one target attribute can be replaced by just one model that predicts all target attributes at once. The task of the second part of experimental evaluation is to investigate this issue.

### 4.1   Data Sets

In order to evaluate the performance of PCRs, we perform experiments on single target and on multiple target problems. For single target problems, we have selected a collection of 15 data sets from the *UCI Machine Learning Repository* [14] which are widely used in various comparative studies.

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 Repository* can also be regarded as multiple target problems (BRIDGES, MONKS, SOLAR-FLARE, and THYROID-0387). In addition, we use the following five data sets.

The EDM is a data set on electrical discharge machining with 154 examples, 16 descriptive attributes and two target attributes. The MEDIANA data set consists of 7953 questionnaires on the Slovene media space, has 79 descriptive attributes and 5 target attributes. The SIGMEA-REAL is a data set on a field study of a genetically modified oilseed rape. It comprises 817 examples, 6 descriptive, and 2 target attributes. The SIGMEA-SIM data set is also concerned with genetically modified oilseed rape, however, the data are produced by a simulation model. The data consists of 10368 examples with 11 descriptive and 2 target attributes. The WATER-QUALITY data set comprises biological and chemical data that were collected through regular monitoring of rivers in Slovenia. The data consists of 1060 examples with 16 descriptive and 14 target attributes.

## 4.2   Evaluation Methodology

When evaluating the newly developed method, we are interested in the predictive error of the learned rule sets and their size, i.e., the number of rules within the rule sets. The CN2 and CN2-WRAcc as well as PCR algorithms can induce ordered or unordered rules, therefore we perform experiments for both. JRip can only learn ordered rules. All error rates are estimated using 10-fold cross-validation. The folds for a specific data set are the same for all experiments. As recommended by [7], significance of the observed differences in error rates and rule set sizes of two algorithms was tested with the *Wilcoxon signed-rank* test.

Unless otherwise noted, all algorithm parameters were set to their default values. 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. We use the *p-value* of 0.99 for significance testing in the CN2 algorithm.

The default parameter values for the PCR algorithm are as follows: beam width $b_w=10$, minimal number of examples $\mu=2$, coverage heuristic weight $\alpha=1$, covering weight $\zeta=0$, and covering weight threshold $\epsilon=0.1$. These are set so as to emulate the CN2 and CN2-WRAcc algorithms as closely as possible and were not tuned in any way. Ordered rules were induced with the learning set modifying method ($M_{mod}$) set to *"Std-Covering"*, while for unordered rules it was set to *"Err-Weight-Covering"*.

The comparison of PCRs used for multiple target prediction and PCRs used for single target prediction is performed as follows. For each data set, we have learned one multiple target PCR model and compared it to a collection of single target PCR models. This collection 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 again estimated using the *Wilcoxon signed-rank* test; each target attribute of each data set corresponds to one data point.

## 5   Experimental Results

### 5.1   Comparison to Existing Methods

First, we present the results of the comparison of predictive clustering rules (PCRs) to the CN2, CN2-WRAcc, and JRip methods. Table 2.a gives the significances of differences for pairs of algorithms for ordered rules and unordered rules. Except for the JRip, we also compared ordered vs. unordered rules. Due to space limits, we have left out the table with complete results for each data set.

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 rules are better in terms of error than ordered PCRs, but the difference is below the significance threshold. Next, if we compare unordered rules, we see that PCRs have a significantly smaller error than the two CN2 algorithms. However, the PCR rule sets are much larger than the CN2-WRAcc rule sets. There is no significant difference between (ordered) JRip

**Table 2.** Significances (p-values) of differences in error rates and rule set sizes for the pairs of algorithms: *CN2, CN2-WRAcc* (CN2W), *JRip*, and *PCR* for ordered (OR) and unordered (UN) rules. 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. **a)** Single target classification, **b)** Single target vs. multiple target classification.

**a)**

| COMPARED ALGORITHMS | | ERROR | | SIZE | |
|---|---|---|---|---|---|
| | | P-VALUE | | P-VALUE | |
| CN2 OR | : CN2W OR | 0.188 | < | **<0.001** | > |
| CN2 OR | : PCR OR | 0.978 | < | 0.151 | > |
| CN2W OR | : PCR OR | 0.359 | > | **0.003** | < |
| JRIP OR | : PCR OR | 0.073 | < | 0.934 | > |
| CN2 UN | : CN2W UN | 0.762 | < | **<0.001** | > |
| CN2 UN | : PCR UN | **0.002** | > | 0.804 | < |
| CN2W UN | : PCR UN | **0.003** | > | **<0.001** | < |
| JRIP OR | : PCR UN | 0.847 | > | **0.007** | < |
| CN2 OR | : CN2 UN | 0.144 | < | 0.359 | < |
| CN2W OR | : CN2W UN | 0.524 | < | 0.804 | > |
| PCR OR | : PCR UN | **0.018** | > | **<0.001** | < |

**b)**

| COMPARED ALGORITHMS | ERROR | | SIZE | |
|---|---|---|---|---|
| SINGLE : MULTIPLE | P-VALUE | | P-VALUE | |
| PCR OR | 0.066 | < | **<0.001** | > |
| PCR UN | 0.067 | > | **<0.001** | > |

rules and unordered PCRs in terms of error, but JRip tends to produce significantly smaller rule sets than the latter. Finally, if we compare ordered and unordered rules induced by each algorithm, 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. From these results, we can conclude that the performance of PCRs on single target problems is comparable to the performance of the CN2 and CN2-WRAcc algorithms for ordered rules, and better for unordered rules. In terms of error, ordered PCRs are somewhat worse than JRip, while unordered PCRs are comparable to JRip.

## 5.2   Comparison of Single to Multiple Target Prediction

The significances of differences between single target and multiple target PCRs are given in Table 2.b, while error rates and rule set sizes are presented in Table 3.

From the Table 2.b we can conclude that ordered multiple target prediction models tend to be less accurate than the single target prediction models. In the case of unordered rules, however, the situation is reversed: multiple target prediction models are better than the single target prediction models. In both cases the difference is almost significant (*p-value* ≃ 0.07). The difference in the rule set sizes, however, is very significant; the size of single target rule sets is roughly twofold in the case of unordered rules, and more than threefold in the case of ordered rules. These results suggest that the multiple target PCRs indeed outperform single target PCRs in terms of rule set size, while the accuracy of both types of models is comparable. In addition, multiple target prediction setting somewhat improves the accuracy of unordered rule sets.

**Table 3.** Comparison of *error rates* of *ordered* (OR) and *unordered* (UN) 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 rule set. In each row, the smallest error rate of ordered and unordered rules is typeset in bold. 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 / TAR. ATT. | PCR OR SINGLE % ERROR | # SIZE | PCR OR MULTIPLE % ERROR | # SIZE | PCR UN SINGLE % ERROR | # SIZE | PCR UN MULTIPLE % ERROR | # SIZE |
|---|---|---|---|---|---|---|---|---|
| BRIDGES | **35.0** | 34 | 40.5 | 7 | 37.3 | 36 | **32.2** | 12 |
| T-OR-D | **19.4** ±14.1 | 4 | 24.7 ±0.0 | | 19.4 ±14.1 | 4 | **10.6** ±8.7 | |
| MATERIAL | 21.6 ±13.5 | 6 | **20.0** ±10.1 | | 26.5 ±17.8 | 6 | **18.8** ±10.4 | |
| SPAN | **31.8** ±14.9 | 6 | 43.5 ±11.3 | | **35.2** ±13.3 | 6 | 40.0 ±11.0 | |
| REL-L | 47.5 ±21.5 | 7 | **44.7** ±0.0 | | 46.5 ±20.6 | 8 | **35.3** ±15.7 | |
| TYPE | **54.9** ±17.2 | 11 | 69.4 ±0.0 | | 58.8 ±12.7 | 12 | **56.5** ±0.0 | |
| EDM | **24.7** | 17 | 25.0 | 9 | **28.2** | 16 | 29.2 | 11 |
| D-FLOW | 13.6 ±15.5 | 7 | **11.7** ±8.1 | | 16.2 ±15.2 | 7 | **12.3** ±9.0 | |
| D-GAP | **35.7** ±11.8 | 10 | 38.3 ±8.2 | | **40.3** ±0.0 | 9 | 46.1 ±13.4 | |
| MEDIANA | 18.2 | 1297 | **17.2** | 271 | 20.1 | 1505 | **16.6** | 685 |
| READ-DELO | 23.1 ±0.9 | 306 | **22.0** ±1.5 | | 24.0 ±1.3 | 353 | **21.7** ±1.2 | |
| READ-DNEVNIK | 21.9 ±1.2 | 436 | **16.6** ±1.4 | | 20.4 ±1.7 | 493 | **15.4** ±0.9 | |
| READ-EKIPA | 9.2 ±0.9 | 296 | **7.1** ±0.8 | | 7.4 ±0.9 | 362 | **6.3** ±0.8 | |
| READ-SL-NOV | **26.3** ±1.4 | 100 | 28.8 ±1.1 | | 38.0 ±4.6 | 100 | **29.2** ±0.9 | |
| READ-VECER | **10.3** ±1.7 | 159 | 11.6 ±0.9 | | 10.5 ±1.1 | 197 | **10.4** ±1.0 | |
| MONKS | **3.3** | 39 | 21.7 | 4 | **17.5** | 40 | 23.5 | 10 |
| MONK-1 | **0.0** ±0.0 | 7 | 30.1 ±9.1 | | **11.1** ±4.9 | 7 | 17.6 ±7.3 | |
| MONK-2 | **10.0** ±6.4 | 28 | 33.1 ±8.0 | | **33.3** ±13.1 | 29 | 35.9 ±5.9 | |
| MONK-3 | **0.0** ±0.0 | 4 | 1.9 ±3.0 | | **8.1** ±7.2 | 4 | 17.1 ±5.4 | |
| SIGMEA-REAL | **24.8** | 76 | 24.9 | 38 | **24.5** | 91 | 24.9 | 72 |
| MFO | 26.1 ±5.4 | 52 | **24.5** ±5.2 | | 26.2 ±5.8 | 60 | **25.1** ±4.6 | |
| MSO | **23.5** ±6.1 | 24 | 25.3 ±3.8 | | **22.8** ±5.3 | 31 | 24.6 ±3.3 | |
| SIGMEA-SIM | **0.7** | 14 | 2.1 | 3 | **1.2** | 15 | 2.1 | 4 |
| DISP-RATE | **1.4** ±0.4 | 12 | 4.3 ±0.7 | | **2.4** ±0.7 | 13 | 4.3 ±0.7 | |
| DISP-SEEDS | **0.0** ±0.0 | 2 | **0.0** ±0.0 | | **0.0** ±0.0 | 2 | **0.0** ±0.0 | |
| SOLAR-FLARE | 11.1 | 58 | **11.0** | 23 | 13.1 | 79 | **10.4** | 39 |
| C-CLASS | 15.8 ±7.6 | 25 | **15.2** ±6.7 | | 18.3 ±8.2 | 36 | **13.6** ±6.9 | |
| M-CLASS | **13.0** ±3.7 | 19 | 14.6 ±4.7 | | 15.8 ±4.0 | 27 | **14.9** ±4.6 | |
| X-CLASS | 4.6 ±4.1 | 14 | **3.1** ±3.2 | | 5.3 ±5.3 | 16 | **2.8** ±3.4 | |
| THYROID-0387 | **1.8** | 666 | 2.4 | 497 | **2.1** | 727 | 2.5 | 560 |
| HYPER-THYRO | **2.0** ±0.5 | 107 | 2.5 ±0.6 | | **1.7** ±0.5 | 115 | 2.5 ±0.5 | |
| HYPO-THYRO | **0.9** ±0.4 | 53 | 3.1 ±0.8 | | **2.4** ±0.7 | 40 | 3.7 ±0.5 | |
| BIND-PROT | **2.9** ±0.8 | 135 | 3.2 ±0.8 | | **3.0** ±0.6 | 157 | 3.4 ±0.6 | |
| GEN-HEALTH | **2.6** ±0.8 | 125 | 3.6 ±0.6 | | 3.0 ±0.7 | 134 | **2.7** ±0.9 | |
| REPL-THEORY | **2.1** ±0.5 | 129 | **2.1** ±0.4 | | **2.2** ±0.7 | 148 | 3.0 ±0.8 | |
| ANTITHYRO-TR | **0.3** ±0.2 | 24 | **0.3** ±0.2 | | **0.4** ±0.2 | 24 | **0.4** ±0.2 | |
| DISC-RESULTS | **1.6** ±0.3 | 93 | 2.0 ±0.5 | | 2.1 ±0.6 | 109 | **2.0** ±0.6 | |

**Table 3.** (*continued*)

| DATA SET<br>TAR. ATT. | PCR OR SINGLE<br>% ERROR # SIZE | | PCR OR MULTIPLE<br>% ERROR # SIZE | | PCR UN SINGLE<br>% ERROR # SIZE | | PCR UN MULTIPLE<br>% ERROR # SIZE | |
|---|---|---|---|---|---|---|---|---|
| WATER-QUALITY | **32.1** | 736 | 33.3 | 89 | 33.9 | 788 | **31.8** | 153 |
| CLAD-SP | **38.5** ±3.7 | 31 | 39.5 ±5.6 | | **39.9** ±4.9 | 28 | 40.4 ±4.9 | |
| GONG-INC | 34.4 ±4.2 | 64 | **29.5** ±3.6 | | 39.2 ±6.6 | 77 | **28.4** ±3.2 | |
| OEDO-SP | 29.8 ±3.6 | 62 | **29.7** ±4.5 | | 30.9 ±4.7 | 78 | **29.9** ±5.1 | |
| TIGE-TEN | 25.1 ±2.6 | 80 | **23.2** ±4.6 | | 23.9 ±4.8 | 83 | **20.8** ±2.4 | |
| MELO-VAR | **34.2** ±4.0 | 30 | 41.7 ±4.8 | | **38.9** ±3.3 | 27 | 41.4 ±3.7 | |
| NITZ-PAL | **29.6** ±3.0 | 23 | 31.3 ±2.3 | | **30.0** ±4.8 | 25 | 30.6 ±4.3 | |
| AUDO-CHA | 30.5 ±5.0 | 88 | **29.3** ±5.0 | | 30.8 ±5.9 | 90 | **24.2** ±5.2 | |
| ERPO-OCT | 31.7 ±3.7 | 75 | **29.0** ±2.6 | | 32.7 ±4.3 | 79 | **26.5** ±3.3 | |
| GAMM-FOSS | **32.3** ±3.8 | 27 | 38.1 ±4.5 | | **33.0** ±5.9 | 23 | 37.8 ±3.7 | |
| BAET-RHOD | 32.0 ±4.8 | 49 | **31.8** ±3.1 | | 33.5 ±5.4 | 57 | **32.5** ±2.4 | |
| HYDRO-SP | **34.9** ±4.5 | 29 | 39.1 ±4.6 | | 39.1 ±4.2 | 33 | **38.2** ±4.9 | |
| RHYA-SP | **29.1** ±4.4 | 59 | 36.1 ±5.5 | | 32.8 ±4.2 | 69 | **30.8** ±6.8 | |
| SIMU-SP | **37.6** ±4.5 | 59 | 38.5 ±5.5 | | 39.3 ±4.6 | 55 | **37.3** ±4.7 | |
| TUBI-SP | 29.2 ±3.8 | 60 | **28.8** ±4.0 | | 31.0 ±3.9 | 64 | **27.1** ±3.1 | |
| AVERAGE | **20.3** | 326.3 | 22.6 | 104.6 | 22.7 | 366.3 | **21.4** | 171.8 |

## 6 Conclusions and Further Work

A new method for learning rules for multiple target classification, called predictive clustering rules, is proposed in this paper. 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 method is empirically evaluated in terms of error and rule set size on several single and multiple target classification problems. First, the method is compared to some existing rule learning methods (CN2, CN2-WRAcc, and JRip) on single target problems. These results suggest that PCRs' performance on single target classification problems is good, and they can be used as a baseline in the next part of the evaluation.

The comparison of multiple target prediction PCRs to the corresponding collection of single target prediction PCRs on multiple target classification problems shows that in the case of ordered rules, the single target prediction models are better, while in 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 new method therefore compares favorably to existing methods on single target problems, while multiple target models (on multiple target problems) offer comparable performance and drastically lower complexity than the corresponding collections of single target models.

Let us conclude with some guidelines for further work. We have only discussed classification problems in this paper. By defining the dispersion measure used in the search

heuristic for numeric attributes, it should be possible to extend the presented algorithm towards regression problems also. Since there are not many methods for learning regression rules, we see this as a worthwhile direction for further research. In addition, there exist several newer methods, e.g., Ripper [6]; incorporating the ideas from these methods into predictive clustering rules could lead to improved performance.

# References

1. Blockeel, H.: Top-down Induction of First Order Logical Decision Trees. PhD thesis, Katholieke Universiteit Leuven, Department of Computer Science, Leuven, Belgium (1998)
2. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the Fifteenth International Conference on Machine Learning, July 1998, pp. 55–63. Morgan Kaufmann, San Francisco (1998)
3. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
4. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In: Proceedings of the Fifth European Working Session on Learning, pp. 151–163. Springer, Berlin (1991)
5. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning 3(4), 261–283 (1989)
6. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
8. Gamberger, D., Lavrač, N.: Expert guided subgroup discovery: Methodology and application. Journal of Artificial Intelligence Research 17, 501–527 (2002)
9. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, New York (1990)
10. Langley, P.: Elements of Machine Learning. Morgan Kaufmann, San Francisco (1996)
11. Lavrač, N., Flach, P., Zupan, B.: Rule evaluation measures: A unifying view. In: Džeroski, S., Flach, P.A. (eds.) ILP 1999. LNCS (LNAI), vol. 1634, pp. 174–185. Springer, Heidelberg (1999)
12. Michalski, R.S.: On the quasi-minimal solution of the general covering problem. In: Proceedings of the Fifth International Symposium on Information Processing (FCIP 1969), Bled, Yugoslavia, Switching Circuits, vol. A3, pp. 125–128 (1969)
13. Michalski, R.S.: 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)
14. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
15. Todorovski, L., Flach, P., Lavrač, N.: Predictive Performance of Weighted Relative Accuracy. In: Zighed, A.D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 255–264. Springer, Heidelberg (2000)
16. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
17. Ženko, B., Džeroski, S., Struyf, J.: Learning Predictive Clustering Rules. In: Bonchi, F., Boulicaut, J.-F. (eds.) KDID 2005. LNCS, vol. 3933, pp. 234–250. Springer, Heidelberg (2006)