Human-like Rule Optimization for Continuous Domains

Fedja Hadzic and Tharam S. Dillon

DEBII, Curtin University of Technology, Perth, Australia
{f.hadzic, t.dillon}@curtin.edu.au

Abstract. When using machine learning techniques for data mining purposes one of the main requirements is that the learned rule set is represented in a comprehensible form. Simpler rules are preferred as they are expected to perform better on unseen data. At the same time the rules should be specific enough so that the misclassification rate is kept to a minimum. In this paper we present a rule optimizing technique motivated by the psychological studies of human concept learning. The technique allows for reasoning to happen at both higher levels of abstraction and lower level of detail in order to optimize the rule set. Information stored at the higher level allows for optimizing processes such as rule splitting, merging and deleting, while the information stored at the lower level allows for determining the attribute relevance for a particular rule. The attributes detected as irrelevant can be removed and the ones previously detected as irrelevant can be reintroduced if necessary. The method is evaluated on the rules extracted from publicly available real world datasets using different classifiers, and the results demonstrate the effectiveness of the presented rule optimizing technique.

Keywords: Data Mining, Rule Optimization, Feature Selection

1 Introduction

Large amounts of data are being collected for different industrial, commercial or scientific purposes, where the aim is to discover new and useful patterns from data that gives rise to discovery of valuable domain knowledge. This process is termed knowledge discovery and the step concerned with applying programs that are capable of learning and generalizing from presented information is called data mining. The end result is a knowledge model that should be easy for human comprehension. This knowledge model is then used by the people involved in that domain for particular domain specific tasks. For example in many businesses it is used as a decision support tool, in medical domains it aids in diagnostic tasks and in more general terms it provides an organization with the basic knowledge of the concepts and their roles and relationships which occur in that particular domain.

The process of using the underlying rules of a knowledge model for classifying future unseen instances is in the data mining field known as the prediction task. The knowledge model can be evaluated based on its predictive accuracy which

corresponds to the percentage of correctly classified instances from an unseen data set. In addition to predictive accuracy, simple rules are preferred since they are more comprehensible and are expected to perform better on unseen data since they are more general. Taking these observations into account, a rule optimizing process needs to make a between the misclassification rate (MR), coverage rate (CR) and generalization power (GP) [1]. MR is measured as the number of incorrectly classified instances while CR is the number of instances that are captured by the rule set. MR should be minimized while the CR should be maximized. Good GP is achieved by simplifying the rules. The trade-off is especially evident when optimizing the rule set from a domain characterized by continuous attributes. An optimal constraint on the attribute range needs to be determined as in many cases increasing the attribute range usually leads to the increase in CR but at the cost of an increase in MR. In regards to obtaining good GP by simplifying the rule, there is a trade-off since if the rules are too general, the MR may increase, since the simple rules may be incapable of distinguishing some more specific cases of the domain. Rule optimization is a type of uncertain reasoning technique and a number of different techniques have been adopted in the literature [1, 2, 3, 4].

The aim of this work is to present a stand-alone rule optimizing technique that is capable of reasoning at the higher level of abstraction as well as lower level of detail. It is an extension of the rule optimizing technique discussed in [5, 6], which was an integral part of the neural network learning method for continuous domains. The reasoning only occurred at the higher level where similar rules were merged and the ones with high MR were split into more specific rules. The lower level instance information was not used and any poor choices made during the network pruning [5] stage could not be corrected. In the proposed extension the lower level information corresponds to the relationships between the values of the defining attributes and the implying class of a rule. This information allows one to determine the relevance of rule attributes at any stage of the rule optimizing (RO) process. Attributes previously found as irrelevant can be re-introduced if they become relevant at a later stage in the process, and at the same time attributes that have lost their predictive capability can be deleted. This corresponds to being able to measure an attributes sequential variation in its predictive capability. This characteristic is very useful for RO since it can often be the case that other attributes may become relevant for more specific distinguishing of a new rule which resulted from splitting of an original rule. On the other hand, some attributes can loose their relevance for predicting the class value implied by a rule, when that rule is obtained through merging of two or more specific rules. Furthermore the method from [5, 6] is only applicable for optimizing of the rules learned by the neural network. The proposed RO technique is applicable to any sets of rules, not only those extracted from a specifically developed system. As such it is capable of incorporating domain expert knowledge which can be represented as a set of rules and which can then be refined and adapted to the future cases. The effectiveness of the proposed method is demonstrated by evaluating it on the rules learned from publicly available real world datasets.

The rest of the paper is organized as follows. In Section 2 we discuss the theoretical motivations of the proposed rule optimizing technique. It also provides an overview of how the theory of concept formation will be mimicked in the proposed method which is described in detail in Section 3. Section 4 provides an experimental

evaluation of the method and some general remarks. The paper is concluded in Section 5.

2 Theoretical Motivation for the Proposed RO Method

From the biological perspective of AI which studies the way humans perform intelligent tasks, a machine learning technique should resemble the way that humans learn. While neural networks resemble this greatly at the brain level of neural interaction, there is also a higher level of reasoning that occurs when a human reasons about the formed knowledge or beliefs. Hence, from the same perspective it would be useful for a rule optimizing technique to resemble the way concept formation and refinement occurs in humans.

The way that humans engage in concept or category formation has been studied extensively in the psychology area. In general terms it corresponds to the process by which a person learns to sort specific observations into general rules or classes, and thereby allows one to respond to events in terms of their class membership rather than uniqueness [7]. It is the elementary form by which humans adjust to their environment. One needs to identify the attributes of relevance for learning or applying a particular rule for formulating a concept [8]. Humans consistently seek confirming information by actively searching they environment for appropriate examples which can confirm or modify the newly discovered concepts [8, 9, 10] Hence, there exists one level at which the concepts or categories have been formed and there is another level where the observations are used for confirming or adjusting the learned concepts and their relationships [11]. When an observation appears to be contradictory to a formed belief, one may engage in thinking at the lower level of detail where the constituents of a belief and the examples that formed it are investigated. Some pre-conditions can be added or removed from the constituents of that particular belief so that the updated belief will not contradict the current observations. In this process of aiming for a reliable belief, it is often the case that features previously found as irrelevant are re-introduced or the ones previously thought to be important are removed from the constituents of a belief. In the context of this work, the term 'belief' corresponds to the rule a human uses to classify the examples or observations into classes or categories.

Performing this type of task is highly desirable for machine learning and our main focus is to allow for such a mechanism in the rule optimizing process. The proposed RO technique is mainly motivated by the psychological studies of human concept formation performed in [7]. In one of the experiments, the human subjects were presented with a number of instances which were classified according to a rule and their task was to discover the rule. In this process, the human subjects formed initial rules from a few observations and then would refine or update these rules when instances contradicting their currently formed rule were observed [7]. This process is simulated in our RO technique since it starts with the initial set of rules and then uses a set of class labelled instances to refine or update the rules according to the instances. The aim is to optimize the rules in such a way that there is an optimal trade-off between misclassification rate, coverage rate and the generalization power. This is

particularly important when continuous attributes are in question since a slight increase in the allowed range leads to a higher coverage rate, but at the same time may increase the misclassification rate. Further, the rules are presented in a structure that is capable of adapting itself according to the future instances. Furthermore we are interested in a structure which can adapt itself to the changes in a domain. The higher level of abstraction would correspond to the rule structure while the low level of detail would correspond to the instance information stored in the structure at the attribute level rather than rule level. So at the higher level we have the rules with the attribute constraints and the predicting class values while at the lower level we have the relationships between attribute values and the occurring class values as detected from the input instances. This information provides the necessary means for determining the relevance of attributes with respect to a particular rule. In other words, we measure the importance of an attribute in predicting the class value implied by a rule. The irrelevant attributes can then be deleted (or re-introduced) at the higher level which will affect the rule coverage rate and generalization power. It is advantageous to integrate a feature selection mechanism since initial bad choices made about the attribute relevance could be corrected as learning progresses.

3 Description of the Proposed RO Technique

This section starts by providing a brief overview of the developed RO technique, and then proceeds to explain each step in more detail in each of the subsequent sections. The general steps of the proposed rule optimization method are presented in the flow chart of Fig. 1. The method takes as input a file describing the rules detected by a particular classifier and the domain dataset from which the rules were learned. As the first step, the rules need to be appropriately represented so as to enable optimization reasoning to occur. The rules are represented in a graph structure (*GS*) where each rule has a set of attribute constraints and a target vector that stores a set of weighted links pointing to one or more target values. Reading the rule set determines the set of attribute constraints for each rule. In order to set the target vector of each rule, the domain dataset is read in on top of the GS triggering those rules whose attribute constraints best match the attribute values in the presented instance from the dataset. Whenever a rule is triggered, its target vector is updated. The target vector of a rule is updated by updating the weight on the link to the target value that occurred in the instance that triggered that particular rule. At this stage, the GS contains the high level information about the domain at hand in the form of rules, their attribute constraints and their target vectors. This information is used for reasoning at the higher level of abstraction. During this process, the rules can undergo a process of splitting, merging and deleting as will be described later in the paper.

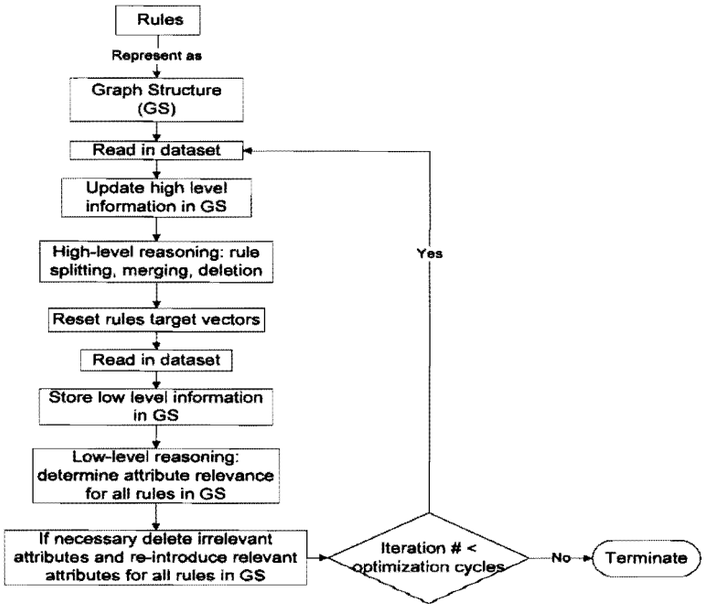

Fig. 1. General steps of the proposed rule optimization method

When two or more rules are merged, it is possible that some attributes have lost their relevance. On the other hand, when a simplified rule is split into two more specific rules, some attributes may become relevant for distinguishing more specific data object characteristics and need to be reintroduced. This is the reason for the reasoning at a lower level that determines the attribute relevance of the rules to occur after the higher level reasoning has been completed. Hence, after the reasoning at the higher level, the target vectors of all the rules in the *GS* are reset and the domain dataset is read in again, this time storing the low level information. The collected information corresponds to the relationships between attribute and target values as detected in the instances from the dataset. This kind of information allows for the calculation of the Symmetrical Tau [12] criterion for determining the relevance of the attributes for a particular rule. Hence, irrelevant attributes can be removed from the rule and attributes previously detected as irrelevant can be re-introduced if the relevance turns out to be sufficiently high throughout the RO process. Each of the RO steps (except for initial rule representation in graph structure) explained in the following subsections repeats for a number of chosen iterations. The following section describes how the rules and related information can be described using a graph structure.

3.1 Formation of the Graph Structure

In order for the *GS* to be formed two files are read, one describing the rules detected by a classifier and the other containing the total set of instances from which the rules were learned. The rules are in form of attribute constraints while the implying class of each rule is ignored. The reason is that during the whole process of *RO*, the implying class values can change as some clusters will be merged or split. Rather the domain dataset is read according to which the weighted links between the rules and class values are set. The implying class value of a rule becomes the highest weighted link to a particular class value node. This class value has most frequently occurred in the instances which were captured by the rule. An example of the *GS* after a dataset is read in is shown in Fig. 2. The implying class of Rule1 and Rule 3 would be class value 1 while for Rule2 it is class value 2. Even though it is not shown in the figure, each rule has a set of attribute constraints associated with it, which we refer to as the weight vector (*WV*) of that rule. The set of attribute values occurring in the instance being processed are referred to as the input vector (*IV*). Hence, to classify an instance we match the *IV* against the *WV*s of the available rules. A constraint for a continuous attribute is given in terms of a lower range (lr) and an upper range (ur) indicating the set of allowed attribute values.

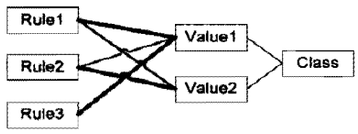

Fig. 2. Example graph structure from high level

3.2 Representing Lower Level Information

Previous sub-section has explained the *GS* formation at the top level which is used mainly for determining the implying class values of the rules. In this section we discuss how lower level instance information is stored for each rule. This low level information is necessary for the reasoning at the lower level.

As previously mentioned each rule has a set of attribute constraints associated with it, which are stored in its *WV*. For each of the attributes in the *WV* we collect the occurring attribute values in the instances that were captured by that particular rule. Hence each attribute has a value list (*VL*) associated with it which stores all the occurring attribute values. Furthermore, each of the value objects in the list has a set of weighted links to the occurring class values in the instance where that particular value occurred. This is necessary for the feature selection process which will be explained later. For a continuous attributes there could be many occurring values and values close to one another are merged into one value object when the difference between the values is less than a chosen merge value threshold. Hence the numerical

values stored in a list of a continuous attribute will be ordered so that a new value is always stored in an appropriate place and the merging can occur if necessary. Fig. 3 illustrates how this low level information is stored for a rule that consists of two continuous attributes A and B. The attribute A has the lower range (lr) and the upper range (ur) in between which the values v1, v2 and v3 occur. The 'lr' of A is equal to the value of v1 or the 'lr; of v1 if v1 is a merged value object, while the 'ur' of A is equal to the value of v3 or the 'ur' of v3 if v3 is a merged value object.

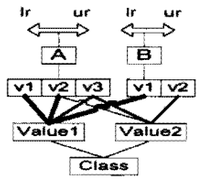

Fig. 3. Storing low level instance information

3.3 Higher Level Reasoning

Once the implying classes are set for each of the rules the dataset is read in again in order to check for any misclassifications and update the rule set accordingly. When a rule captures an instance that has a different class value than the implication of the rule, a child rule will be created in order to isolate the characteristic of the rule causing the misclassification. The attribute constraints of the parent and child rule are updated so that they are exclusive from one another. The child attribute constraint ranges from the attribute value of the instance to the range limit of the parent rule to which the input attribute value was closest to. The parent rule adopts the remaining range as the constraint for the attribute at hand.

To illustrate the process of making an attribute constraint exclusive from one another in the parent and child rule, please consider Fig. 4. At the top of the figure we show the lower (*LR*) and upper range (*UR*) of an attribute a_k occurring in the weight vector of the parent rule (r_s) at position *k* and an input value (*IVk)* that has occurred within the range. In this example, the value *IVk* was closer to the upper limit of the range. The newly created attribute constraint for the child rule (bottom right of Fig. 4) will have its lower range (i.e. *LR'*) set to value of *IVk* and the upper range (i.e. *UR'*) is equal to UR. The constraint in the parent rule (bottom left of Fig. 4) is updated so that the new upper range is set to be a small value (i.e. *smlVal*) away from *IVk* . The process for all the cases is more formally explained below.

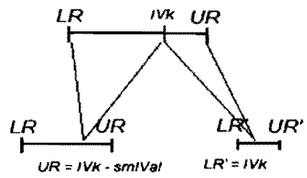

Fig. 4. Illustrating the update of range constraints for an attribute a_k of parent and child rule

After the whole dataset is read in there could be many child rules created from a parent rule. Some child rules may be merged together first but explanation of this is to come later once we discuss the process of rule similarity comparison and merging. If a child rule points to other target values with high confidence it become a new rule and this corresponds to the process of rule splitting, since the parent rule has been modified to exclude the child rule which is now a rule on its own. On the other hand if the child rule still mainly points to the implying class value of the parent rule it is merged back into the parent rule (if they are still similar enough). An example of a rule which has been modified to contain a few children due to the misclassifications is displayed in Fig. 5. The reasoning explained would merge 'Child3' back into the parent rule since it points to the implying class of the parent rule with high weight. This is assuming that they are still similar enough. On the other hand Child1 and Child2 would become new rules since they more frequently capture the instances where the class value is different to the implying class of the parent rule. Furthermore if they are similar enough they would be merged into one rule.

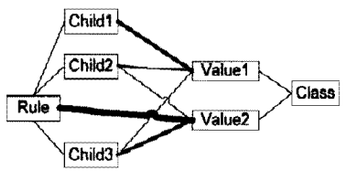

Fig. 5. Example of rule splitting

In order to measure the similarity among the rules we make use of a modified Euclidean distance (*ED*) measure. This measure is also used to determine which rule captures a presented instance. An instance is always assigned to the rule with the smallest *ED* to the *IV*. Even though one would expect the *ED* to be equal to 0 when classifying instances this may not always be the case throughout the *RO* process. The *ED* calculation is calculated according to the difference in the allowed range values of a particular attribute. The way that *ED* is calculated is what determines the similarity among rules, and therefore we first overview the *ED* calculation and then proceed onto explaining the merging of rules that may occur in the whole *RO* process.

3.3.1 Euclidean Distance Calculation

For a continuous attribute a_i occurring at the position i of WV of rule R, let 'a_ilr' denote the lower range, 'a_iur' the upper range, and 'a_iv' the initial value if the ranges of a_i are not set. The value from the i-th attribute of IV will be denotes as iva_i. The i-th term of the ED calculation between IV and WV of R for continuous attributes is:

- case 1: a_i ranges are not set
  - 0 iff $iva_i = a_iv$
  - $iva_i - a_iv$ if $iva_i > a_iv$
  - $a_iv - iva_i$ if $iva_i < a_iv$
- case 2: a_i ranges are set
  - 0 iff $iva_i \geq a_ilr$ and $iva_i \leq a_iur$
  - $a_ilr - iva_i$ if $iva_i < a_ilr$
  - $iva_i - a_iur$ if $iva_i > a_iur$

The input merge threshold used for continuous attribute (MT) also needs to be set with respect to the number of continuous attributes in the set. It corresponds to the maximum allowed sum of the range differences among the WV and IV so that the rule would capture the instance at hand.

When calculating the ED for the purpose of merging similar rules there are four possibilities that need to be accounted with respect to the ranges being set in the rule attributes, and the ED calculation is adjusted. For rule R_1 let r_1a_i denote the attribute occurring at the position i of WV of rule R_1, let 'r_1a_ilr' denote the lower range, 'r_1a_iur' the upper range, and 'r_1a_iv' the initial value if the ranges of r_1a_i are not set. Similarly for rule R_2 let r_2a_i denote the attribute occurring at the position i of WV of rule R_2, let 'r_2a_ilr' denote the lower range, 'r_2a_iur' the upper range, and 'r_2a_iv' the initial value if the ranges of r_2a_i are not set. The i-th term of the ED calculation between the WV of R_1 and WV of R_2 for continuous attributes is:

- case 1: both r_1a_i and r_2a_i ranges are not set
  - 0 iff $r_1a_iv = r_2a_iv$
  - $r_1a_iv - r_2a_iv$ if $r_1a_iv > r_2a_iv$
  - $r_2a_iv - r_1a_iv$ if $r_1a_iv < r_2a_iv$
- case 2: r_1a_i ranges are set and r_2a_i ranges are not set
  - 0 iff $r_2a_iv \geq r_1a_ilr$ and $r_2a_iv \leq r_1a_iur$
  - $r_1a_ilr - r_2a_iv$ if $r_2a_iv < r_1a_ilr$
  - $r_2a_iv - r_1a_iur$ if $r_2a_iv > r_1a_iur$
- case 3: r_1a_i ranges are not set and r_2a_i ranges are set
  - 0 iff $r_1a_iv \geq r_2a_ilr$ and $r_1a_iv \leq r_2a_iur$
  - $r_2a_ilr - r_1a_iv$ if $r_1a_iv < r_1a_ilr$
  - $r_1a_iv - r_2a_iur$ if $r_1a_iv > r_2a_iur$
- case 4: both r_1a_i and r_2a_i ranges are set
  - 0 iff $r_1a_ilr \geq r_2a_ilr$ and $r_1a_iur \leq r_2a_iur$
  - 0 iff $r_2a_ilr \geq r_1a_ilr$ and $r_2a_iur \leq r_1a_iur$
  - $\min(r_1a_ilr - r_2a_ilr, r_1a_iur - r_2a_iur)$ iff $r_1a_ilr > r_2a_ilr$ and $r_1a_iur > r_2a_iur$

- $\min(r_2a_ilr - r_1a_ilr, r_2a_iur - r_1a_iur$ iff $r_2a_ilr > r_1a_ilr$ and $r_2a_iur > r_1a_iur$
- $(r_1a_ilr - r_2a_iur)$ iff $r_1a_ilr > r_2a_iur$
- $(r_2a_ilr - r_1a_iur)$ iff $r_2a_ilr > r_1a_iur$

For a rule to capture an instance or for it to be considered sufficiently similar to another rule the *ED* would need to be smaller than the *MT* threshold.

3.3.2 Merging of Similar Rules

As mentioned at the start of Section 3.3 the child rules may be created when a particular rule captures an instance that has a different class value than the implying class value of that rule (i.e. misclassification occurs). After the whole file is read in the child rules that have the same implying class values are merged together if the *ED* between them is below the *MT*. Thereafter the child rules either become a new rule or are merged back into the parent rule, as discussed earlier. Once all the child rules have been validated the merging can occur among the new rule set. Hence if any of the rules have the same implying class value and the *ED* between them is below the *MT* the rules will be merged together and the attribute constraints updated. After this process the file is read in again and any of the rules that do not capture any instances are deleted from the rule set.

3.4 Lower Level Reasoning

Once the rules have undergone the process of splitting and merging, the relevance of rule attributes should be calculated as some attributes may have lost their relevance through merging of two or more rules. Other attributes may have become relevant as a more specific distinguishing factor of a new rule which resulted from splitting of an original rule. For this purpose we make use of the Symmetrical Tau [12] feature selection criterion whose calculation is made possible by the information stored at the lower level of the graph structure. We start this section by discussing the properties of the symmetrical tau and then proceed onto explaining how the relevance cut-off is determined and the issue of choosing the merge value threshold for the value objects in a value list.

3.4.1 Feature Selection Measure

Symmetrical Tau (τ) [12] is a statistical measure for the capability of an attribute in predicting the class of another attribute. The τ measure is calculated using a contingency table which is used in statistical area to record and analyze the relationship between two or more variables. If there are *I* rows and *J* columns in the table, the probability that an individual belongs to row category i and column category j is represented as *P(ij)*, and *P(i+)* and *P(+j)* are the marginal probabilities in row category i and column category j respectively, the Symmetrical Tau measure is defined as [12]:

$$\tau = \frac{\sum_{j=1}^{J}\sum_{i=1}^{I}\frac{(Pij)^2}{P(+j)} + \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{P(ij)^2}{P(i+)} - \sum_{i=1}^{I}P(i+)^2 - \sum_{j=1}^{J}P(+j)^2}{2 - \sum_{i=1}^{I}P(i+)^2 - \sum_{j=1}^{J}P(+j)^2}$$

For the purpose of feature selection problem one criteria in the contingency table could be viewed as an attribute and the other as the target class that needs to be predicted. In our case the information contained in a contingency table between the rule attributes and the class attributes is stored at the lower level of the graph structure as explained in Section 3.2. The τ measure was used as a filter approach for the feature subset selection problem in [13]. In the current work its capability of measuring the sequential variation of an attribute's predictive capability is exploited.

3.4.2 Calculating Relevance Cut-off

For each of the rules that are triggered for multiple class values we calculate the τ criterion and rank the rule attributes according to the decreasing τ value. The relevance cut-off point is determined as the point in the ranking where the τ value of an attribute is less than half of the previous attribute's τ value. All the attributes below the cut-off point are considered irrelevant for that particular rule and are removed from the rule's *WV*. On the other hand, if some of the attributes above the relevance cut-off point were previously excluded from the *WV* of the rule, they are now re-introduced since their τ value indicates their relevance for the rule at hand.

As mentioned in Section 3.2 when the occurring values stored in the value list of an attribute are close together they are merged and the new value object represents a range of values. The merge value threshold chosen determines when the difference among the value objects is sufficiently small for merging to occur. This is important for appropriate τ calculation. Ideally a good merge value threshold will be picked with respect to the value distribution of that particular attribute. However, this information is not always available and in our approach we pick a general merge threshold of around 0.02. This has some implications for the calculated τ value since when the categories of an attribute A are increased more is known about attribute A and the error in predicting attribute B may decrease. Hence, if the merge value threshold is too large many attributes will be considered as irrelevant since all the occurring values could be merged into one value object which points to many target objects and this aspect would indicate no distinguishing property of the attribute. On the other hand, if it is too small many value objects may exist which may wrongly indicate that the attribute has high relevance in predicting the class attribute.

3.4 Summary of the Method

The whole set of RO processes is usually repeated for around 10 iterations. Each time a new iteration is started, the dataset is read in so that the implying class values of the

rules can be determined. The dataset is read in again during which misclassifications are detected, and the rules where the misclassifications occur are split in order to isolate the characteristic which leads to the wrong prediction of the class value. The child rules which have the same implying class value and are similar according to the *ED* are merged together. Thereafter, the child rules are either merged back into the parent rule or become a new rule if they are not similar to the parent rule with respect to the implying class value and the *ED* between their weight vectors. The whole rule set is then traversed to merge any further similar rules. The dataset is then read in to store the lower level instance information according to which the τ criterion can be calculated and irrelevant attributes deleted for a rule, and relevant ones re-introduced when necessary. After a number of iterations, the unseen test file is used to determine the predictive accuracy of the optimized rule set.

4 Method Evaluation

The proposed method was evaluated on two rule sets learned from publicly available real world datasets [14]. The rule optimizing process was run for 10 iterations for each of the tested domains. The first set of rules we consider has been learned from the 'Iris' dataset using the continuous self-organizing map [5] so that we can compare the improvement of the extension to the rule optimizing method. The merge cluster threshold *MT* was set to 0.1 and the merge value threshold *MVT* for attribute values was set to 0.02. The rules obtained using the CSOM technique [5] are displayed in Fig. 6. When the rules obtained after retraining were taken as input by our proposed rule optimization method the resulting rule set was different in only one rule. The rule 4 was further simplified to exclude the attribute constraint from sepal-width and the new attribute constraint was only that petal-width has to be between the values of 0.667 and 1.0 for the class value of Iris-virginica. Hence the process was able to detect another attribute that has become irrelevant during the *RO* process. The predictive accuracy remained the same.

C#	Constraints	TargetVector	MR
1	0.24 < SL < 0.694 0.08 < SW < 0.58 0.42 < PL < 0.695 0.36 < PW < 0.62	Ivs – 26 Ivg – 1	1
2	0.41 < SL < 0.556 0.2 < SW < 0.49 0.64 < PL < 0.78 0.54 < PW < 0.75	Ivg – 9	
3	0 < PW < 0.167	Is – 33	
4	0.25 < SW < 0.75 0.75 < PW < 1	Ivg – 21	
5	0.194 < SL < 0.22 0.37 < PW < 0.41	Ivs – 2	
6	0.375 < SW < 0.5 0.625 < PW < 0.7	Ivg – 5 Ivs – 1	1

C#	Constraints	TargetVector	MR
1	0.24 < SL < 0.75 0.08 < SW < 0.58 0.42 < PL < 0.695 0.36 < PW < 0.62	Ivs – 17 Ivg – 1	1
2	0.41 < SL < 0.556 0.2 < SW < 0.49 0.64 < PL < 0.78 0.54 < PW < 0.75	Ivg – 1	
3	0 < PW < 0.208	Is – 15	
4	0.2 < SW < 0.75 0.667 < PW < 1	Ivg – 12	
5	0.19 < SL < 0.22 0.37 < PW < 0.41	Ivs – 1	
6	0.375 < SW < 0.5 0.625 < PW < 0.7	Ivs – 1	

Fig. 6. Iris rule set as obtained by using the traditional rule optimizing technique

With respect to using CSOM to extract rules from the 'Iris' domain we have performed another experiment. The initial rules extracted by CSOM without the network pruning and retraining of the network were optimized. When network pruning occurs the network should be re-trained for new abstractions to be properly formed. In this experiment we wanted to see how the *RO* technique performs by itself without any network pruning or retraining.

Rules	Implying class
0.33 < PL < 0.678 0.375 < PW < 0.792	Iris-versicolor
0.208 < SW < 0.542 0.627 < PL < 0.847 0.54 < PW < 1.0	Iris-virginica
0.778 < SL < 1.0 0.25 < SW < 0.75 0.814 < PL < 1.0 0.625 < PW < 0.917	Iris-virginica
0.0 < SL < 0.417 0.41 < SW < 0.917 0.0 < PL < 0.153 0.0 < PW < 0.208	Iris-setosa

Fig. 7. Optimized initial rules extracted by CSOM Notation: SL – sepal_length, SW – sepal_width, PL – petal _length, PW – petal_width

By applying the *RO* technique the rule set was reduced to four rules as displayed in Fig. 7. However, not as many attributes were removed from each of the rules and two instances were misclassified. Hence, performing network pruning and retraining prior to *RO* may achieve a more optimal rule set. However, in the cases where retraining the network may be too expensive the *RO* technique can be applied by itself. In fact compared to the initial set of rules detected by CSOM, which consisted of nine rules with three misclassified instances this is still a significant improvement.

The second set of experiments was performed on the complex 'Sonar' dataset which consists of sixty continuous attributes. The examples are classified into two groups one identified as rocks (R) and the second identified as metal cylinders (M). The learned decision tree by the C4.5 algorithm [15] consisted of 18 rules with the predictive accuracy equal to 65.1%. These rules were taken as input in our *RO* technique and the *MT* was set to 0.2 while the *MVT* was set to 0.0005. The optimized rule set consisted of only two rules i.e 0.0 < a11 <= 0.197 → R and 0.197 < a11 <= 1.0 → M. When tested on an unseen dataset the predictive accuracy was 82.2 % i.e. 11 instances were misclassified from the available 62. Hence the *RO* process has again proved useful in simplifying the rules set without the cost of increasing the number of misclassified instances..

5 Conclusions

This work has described a rule optimizing technique suitable for domains characterized by continuous attributes. It is applicable to the optimization of rules obtained from any data mining techniques, and with such a characteristic it allows for

the incorporation of domain expert knowledge. This domain knowledge can be represented as a set of rules, which are then to be fine tuned according to the newly collected data from the domain. Hence the method as a whole is adaptable to the changes in the domain it is applied to. Being able to swap from higher level reasoning to the reasoning at the lower instance level has indeed proven useful for determining the relevance of attributes throughout the rule optimizing process. The evaluation of the method on the rules learned from real world data by different classifier methods has shown its effectiveness in optimizing the rule set. As a future work method needs to be extended so that categorical attributes can be handled as well. Furthermore, it would be interesting to explore the possibilities of the rule optimizing method in becoming a stand-alone machine learning method itself.

References

1. Wang, D.H., Dillon, T S., Chang, E.: Trading off between misclassification, recognition and generalization in data mining with continuous features. In: Developments in Applied Artificial Intelligence (Proc. of the 15th Int'l Conf. on Industrial & Engineering Application of Artificial Intelligence & Expert Systems, *Lecture Notes in Artificial Intelligence*, LNAI 2358, Springer, T. Hendtlass and M. Ali (eds)., June 17-20, pp.303-313, Cairns, (2002).
2. Abe, S. & Sakaguchi, K.: Generalization Improvement of a Fuzzy Classifier with Ellipsoidal Regions. In: Proceedings of the 10th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2001), pp. 207-210, Melbourne, (2001)
3. Chen, Z.: Data mining and uncertain reasoning: an integrated approach. John Wiley & Sons, Inc., New York, (2001).
4. Engelbrecht, A.P.: Computational intelligence : an introduction.. J. Wiley & Sons, Hoboken, New Jersey, (2002).
5. Hadzic, F., Dillon, T.S.: CSOM: Self Organizing Map for Continuous Data. In: *3rd International IEEE Conference on Industrial Informatics (INDIN'05)*, 10-12 August, Perth, (2005).
6. Hadzic, F., Dillon, T.S.: CSOM for Mixed Data Types, In: *Fourth International Symposium on Neural Networks*, June 3-7, Nanjing, China, (2007).
7. Bruner, J.S., Goodnow, J.J., Austin, G.A.: A study of thinking. John Wiley & Sons, Inc., New York, (1956).
8. Sestito, S., Dillon, S.T.: Automated Knowledge Acquisition. Prentice Hall of Australia Pty Ltd, Sydney, (1994).
9. Kristal, L.: ABC of Psychology (ed.). Michael Joseph, London, pp. 56-57, (1981).
10. Pollio, H.R.: The psyehology of Symbolic Activity. Addison-Wesley, Reading, Massachusetts, (1974).
11. Roch, E.: Classification of real-world objects: Origins and representations in cognition. in Thinking: Readings in Cognitive Seienee, (eds) P.N. Johnson-Laird & P.C. Wason, Cambridge University Press, Cambridge, pp. 212-222, (1977).
12. Zhou, X., Dillon, T.S.: A statistical-heuristic feature selection criterion for decision tree induction. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no.8, August, pp 834-841, (1991).
13. Hadzic, F, Dillon, T.S.: Using the Symmetrical Tau (τ) Criterion for Feature Selection in Decision Tree and Neural Network Learning. In: Proceedings of the 2nd Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics, in conjunction with the 2006 SIAM International Conference on Data Mining April 22, Bethesda, (2006).
14. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases, Irvine, CA: University of California, Department of Information and Computer Science., (1998). [http://www.ics.uci.edu/~mlearn/MLRepository.html].
15. Quinlan, J.R.: Probabilistic Decision Trees. Machine Learning: An Artificial Intelligence Approach Volume 4, Kadratoff, Y & Michalski, R., Morgan Kaufmann Publishers, Inc., San Mateo, California, (1990).