Using Competitive Learning between Symbolic Rules as a Knowledge Learning Method

F. Hadzic[1] and T.S. Dillon[2]

Abstract We present a new knowledge learning method suitable for extracting symbolic rules from domains characterized by continuous domains. It uses the idea of competitive learning, symbolic rule reasoning and it integrates a statistical measure for relevance analysis during the learning process. The knowledge is in form of standard production rules which are available at any time during the learning process. The competition occurs among the rules for capturing a presented instance and the rules can undergo processes of merging, splitting, simplifying and deleting. Reasoning occurs at both higher level of abstraction and lower level of detail. The method is evaluated on publicly available real world datasets.

1 Introduction

Within the AI community there are two views on how a machine should engage in the knowledge learning task. Some believe that the human learning should be mimicked in the way that it is initially learned at the lower neuronal level (connectionism) while others believe it is the higher level conceptual reasoning with symbolic rules (symbolism) that will bring machine learning closer to human learning [1]. Anderson [2] believes that both neural aspects and cognitive rules exist and that it is unlikely to have an accurate psychological theory without the formulation of explicit rules that enable proper representation of the acquired generalizations. Chandasekaran et al. [3] stated that connectionism and symbolism

[1] F. Hadzic

Digital Ecosystems and Business Intelligence Institute, GPO Box U1987 Perth, Australia email: fedja.hadzic@postgrad.curtin.edu.au

[2] Prof. T.S.Dillon

Digital Ecosystems and Business Intelligence Institute, GPO Box U1987 Perth, Australia email: tharam.dillon@cbs.curtin.edu.au

Please use the following format when citing this chapter:

Hadzic, F. and Dillon, T.S., 2008, in IFIP International Federation for Information Processing, Volume 276; *Artificial Intelligence and Practice II*; Max Bramer; (Boston: Springer), pp. 351360.

both agree on the view of intelligence as information processing of representations, but disagree on the storage and processing mechanism of those representations. These observations, together with the advantages and disadvantages of symbolic and sub-symbolic systems, justifies the large amount of research that has gone into the extraction of symbolic knowledge from neural networks.

In [4] we presented a variation of the Self-Organizing Map (SOM) [5], CSOM, that applies a different learning mechanism useful for situations where the aim is to extract rules from a data set characterized by continuous input features. Attribute constraints for continuous attributes are contained on the network links themselves, thereby making the network itself more symbolic. A rule optimizing method was integrated into CSOM and in [6] it was adjusted so that it can be applied to rule sets obtained using other knowledge learning methods. Its potential of optimizing rules set through symbolic reasoning has motivated us to investigate whether the method itself can be extended so that it becomes a stand-alone knowledge learning method.

The aim of the work presented in this paper is to present a new knowledge learning method that combines symbolic and sub-symbolic reasoning, but is capable of presenting its acquired knowledge at any time with symbolic rules. It is motivated partly by the competitive learning and progressive parameter adjustment that occurs in the SOM [5]. The main difference is that the competition occurs among the rules for capturing the instances rather than neurons and hence symbolic rules are available at any time during learning. The system is able to move from the lower level where weight interactions are used for reasoning to the higher level where symbolic rules are represented and reasoned with. The proposed method can be viewed as an intersection of competitive learning, symbolic reasoning and statistics.

2 Method Description

The dataset describing the domain at hand is split into an unsupervised (class labels removed) and supervised (class labels present) dataset. Initially the rule set is empty and the unsupervised file is used according to which the initial rule set is obtained. The main part according to which the learning is done is the structure used to represent the currently learned rules which are in form of antecedent-consequent pairs. Initially, the rule set is empty and as the instances from a file are read in rules are set up whose antecedents are equal to the attribute values occurring in the samples. When a set of instances is fed on top of the rule set, it can be said that competition occurs between the rules for capturing the presented instances. The rule that most closely matches the instance in terms of attribute constraints captures that instance and its attribute constraints are adjusted if necessary. During this process similar rules may be merged while contradicting

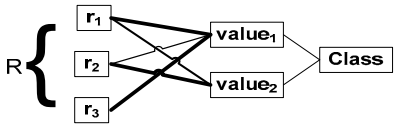
rules may be split. The merging of rules is highly dependent on the threshold chosen according to which two rules are considered similar in terms of attribute constraints. Two learning parameters are used which are progressively adjusted during the learning. One parameter is for allowing an instance to be captured by the rule (*IR*) and the second is for allowing two rules to merge together (*MR*). Setting both of these parameters to very low values initially will result in a large rule set. As learning proceeds, both parameters are increased. By increasing the *IR* the set of rules will capture more instances thereby increasing the coverage rate of the rules and decreasing the number of new rules that need to be formed. By increasing the *MR* similar rules are more likely to merge into one making the rule set more general and reducing the number of rules. Splitting of rules occurs when a misclassification occurs. Similarly the instance information about attribute and class relationships is collected which allows for the reasoning to occur at the lower level (sub-symbolic). At this level the relevance of attributes for a particular rule is determined using a statistical measure. The process as a whole is repeated for a chosen number of iterations.

2.1 Representative Structure

Let m denote the number of input attributes (denoted as x_i) in a dataset D from which the knowledge is to be learned. An instance or example e from D will be referred to as an input vector and denoted by $IV_e = (x_1, x_2, \ldots, x_m)$. Further let x_t denote the class value associated with example e. Let R denote the set of rules in the structure. A rule will be generally denoted as r_p, where $p = (1, \ldots, |R|)$. The attribute constraints of each rule (i.e. the antecedent part of the rule) are contained in the weight vector of that rule (denoted as $WV(r_p)$). An attribute constraint at position i in $WV(r_p)$ is denoted as a_i. Even though some rules will not contain the total set of attributes in their WV the ordering is kept so that items at the same index positions in the IV and the WV correspond to the values or constraints of same attributes in the dataset (i.e. $x_i = a_i$). Hence, to classify an instance e we match the IV_e against the weight vectors of the available rules using a modification of the Euclidean distance (*ED*) measure as the basis for comparison. This process is explained in detail later in the paper. Each rule r_p has a target vector associated with it denoted as $TV(r_p)$ which contains the links to class values that have occurred in the instances that are covered by that particular rule. Let an item at position t in $TV(r_p)$ be denoted as tv_t and the weight associated with it as $w(tv_t)$. The implying class value of a rule r_p becomes the one which has the highest weight associated with it. In other words if the implying class corresponds to the item in $TV(r_p)$ at position x then $\max(w(tv_t)) = w(tv_x) \; \forall \; t \mid t = (1, \ldots, |TV(r_p)|)$. This class value has most frequently occurred in the instances that were captured by the rule. An example of the structure representing the rule set with target vector information is shown in Figure 1 (thicker lines correspond to higher weights on links). The implying class of rules r_1 and r_3 would be class value1 while for rule r_2

it is class value2. Even though it is not shown in Figure 1, the attribute constraints associated with each rule are stored in the *weight vector* of that rule.

Figure 1: Example structure representing the rule set and related information.



2.1.1 Storing Low-Level Information

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. For each $a_i \in WV(r_p)$, where $i = (1, \ldots, m)$ the occurring attribute values in the instances that were captured by that particular rule are stored in a value list (denoted as $VL(a_i)$). The items in $VL(a_i)$ are referred to as value objects and an item at position r in $VL(a_i)$ is denoted as v_r. Each $v_r \in VL(a_i)$, where r = (1,…, $|VL(a_i)|$) has a target vector associated with it, denoted as $TV(v_r)$ which contains the links to class values that have occurred together with that particular value in the instance captured by the rule.

Since for continuous attributes there could be many occurring values, close values are merged into one value object when the difference between the values is less than a chosen merge value threshold. Hence a value object v_r may either contain a singular value, denoted as v_{rVAL} or a lower limit and the upper limit on the range of values, denoted as v_{rLR} and v_{rUR}, respectively. If an attribute value at position i in IV_e is denoted as x_i and the target or class value as x_t then the update process can be summarized as follows. When a rule r_p captures an instance then a link to x_t is added to $TV(r_p)$ with weight set to 1, or the weight on the link is incremented if $TV(r_p)$ already contains a link to that particular x_t. For each $a_i \in WV(r_p)$ the $VL(a_k)$ is updated by either incrementing the weight of a v_r if $x_i = v_{rVAL}$ or $v_{rLR} <= x_i <= v_{rUR}$ *if* v_r ranges are set or otherwise adding a new v_r ($v_{rVAL} = x_i$) (i.e. inserting x_i value in position r in $VL(a_i)$) such that $v_{(r-1)VAL} < v_{rVAL} < v_{(r+1)VAL}$ or if the $v_{(r-1)}$ and $v_{(r+1)}$ are ranged value objects then $v_{(r-1)UR} < v_{rVAL} < v_{(r+1)LR}$. Furthermore, a link to x_t is added to $TV(v_r)$ with weight set to 1, or the weight on the link is incremented if $TV(v_r)$ already contains a link to that particular x_t. Hence the numerical values stored in *VL* of an attribute will be ordered so that a new value is always stored in an appropriate place and the merging can occur if necessary.

Figure 2 illustrates how this low level information is stored for a rule that consists of two continuous attribute a_1 and a_2, and points to two class values (i.e. Value1 and Value2). For example the attribute a_1 has the lower range and the upper range in between which the values v_1, v_2 and v_3 have occurred. The lower range of a_1 is equal to v_{1VAL} or the v_{1LR} if v_1 is a merged value object, while the upper range of a_1 is equal to v_{3VAL} or the v_{3UR} if v_3 is a merged value object.

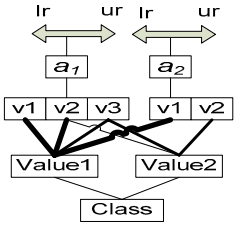


Figure 2: Storing low level instance information.

2.2 Measure for Capturing Instances and Rule Merging

To determine which particular rule should capture an instance and whether two rules should be merged we make use of a modified Euclidean distance (*ED*) measure. An instance *e* is always captured by the rule with the smallest *ED* between the rule's weight vector and the IV_e. The notation used for measuring *ED* between IV_e and a *WV* of a rule r_p is $ED(IV_e, WV(r_p))$. Similarly if we are measuring similarity among two rules r_1 and r_2 then the notation $ED(WV(r_1), WV(r_2))$ is used. The i-th term of $ED(IV, WV(r_p))$ is equal to the distance of the input attribute value x_i to the nearest range boundary of the i-th item in $WV(r_p)$.

Let the i-th term of the $ED(IV_e, WV(r_p))$ be denoted by $r_p term_i$. To determine which rule most closely matches the *IV* the following expression is used.

$$\arg\min\left(\sqrt{\sum_{i=1}^{n}\left(r_p term_i\right)^2}\right)$$

$$\forall\, p \mid p = \{1 \ldots |R|\}.$$

The *IR* needs to be set with respect to the number of attributes in the dataset. It corresponds to the maximum allowed sum of the range/value differences among the attributes of $WV(r_p)$ and IV_e so that the rule would capture the instance at hand. The instance is captured by the rule with the smallest *ED* between its weight vector and the IV_e. If no rule exactly matches IV_e *(i.e.* $ED(IV_e, WV(r_p)) \neq 0 \;\forall$ p = (1, …, $|R|$)) and no rule is close to the instance (i.e. $ED(IV_e, WV(r_p)) >$ *InstToRuleThr* $\forall$ p = (1, …, $|R|$)) then a new rule r_n will be created where all attribute values in its weight vector are set according to the instance attribute values (i.e. $a_i v = x_i \;\forall$ i = (1, …, *m*)).

When calculating the *ED* for the purpose of merging similar rules there are four possibilities that need to be accounted for with respect to the ranges being set in the rule attributes. Two rules r_1 and r_2 will be merged if the $ED(WV(r_1), WV(r_2)) <$ *MR*. For rule r_1 let $r_1 a_i$ denote the attribute occurring at the position *i* of $WV(r_1)$, let $r_1 a_i lr$ denote the lower range, $r_1 a_i ur$ the upper range, and $r_1 a_i v$ the initial value if the ranges of $r_1 a_i$ are not set. Similarly for rule r_2 let $r_2 a_i$ denote the attribute occurring at the position i of $WV(r_2)$, let $r_2 a_i lr$ denote the lower range, $r_2 a_i ur$ 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(WV(r_1), WV(r_2))* calculation 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 $r_2a_ilr > r_1a_iur$

2.3 Reasoning Process

This section explains the reasoning processes that occurs with the information stored in the knowledge structure as explained in the previous sections.

2.3.1 Higher Level Reasoning

Once the implying classes are set for each of the rules, the dataset is fed on topf of the rules. When a rule captures an instance that has a different class value than the implication of the rule, a child rule is 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. In other words, in future an instance will be either captured by the parent rule or the child rule, not both. After the whole dataset is read in, there could be many child rules created from a parent rule. If a child rule points to other target values with high confidence it becomes a new rule. 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. An example of a rule for which a number of children were created due to the misclassifications is displayed in Figure 3. The reasoning explained would

merge 'Child3' back into the parent rule, while Child1 and Child2 would become new rules.

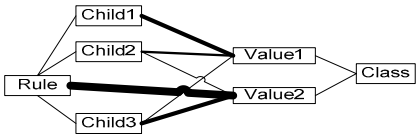


Figure 3: Example of rule splitting.

Rule Merging. 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 *MR*. Once all the child rules have been validated the merging can occur in the new rule set. Hence if two rules r_1 and r_2 have the same implying class value and the $ED(WV(r_1), WV(r_2)) < MR$ the rules will be merged together and the attribute constraints updated. Rather than creating a new rule at the implementation level the merged rule is one of the original rules (say r_1) with its weight vector updated accordingly, while the second rule is removed from the rule set. The update is done in such manner so that the attribute constraints in the $WV(r_2)$ are contained within the attribute constraints of $WV(r_1)$ *(i.e.* $ED(WV(r_1), WV(r_2)) = 0$). Hence, if the range of allowed values for an attribute in the $WV(r_2)$ fell outside the corresponding attribute value range in $WV(r_1)$ then that particular range is expanded to include the value range in $WV(r_2)$. More formally the process can be explained as follows.

The same notation as earlier will be used where the items occurring at position i in $WV(r_1)$ and $WV(r_2)$ will be denoted as r_1a_i and r_2a_i, respectively. If the ranges on the items are not set, then the same notation r_1a_i and r_2a_i also corresponds to the initial values of those attributes. Otherwise $_{LR}$ or $_{UR}$ is appended for lower range or upper range respectively (eg. r_sa_{iLR}). Depending on whether the range of the items in the weight vector is set, there are four cases which determine in which way the $WV(r_1)$(the new merged rule) is updated. It can be expressed as follows. In the logic expressed below the cases when no update needs to occur were excluded. These are the cases when the value(s) of r_2a_i was either equal to the value of r_1a_i or if ranges on r_2a_i are set they fell within the ranges of r_1a_i.

Case 1: r_1a_i and r_2a_i ranges are not set

```
If r1ai  > r2ai
  r1aiUR = r1ai
  r1aiLR = r2ai
If r1ai  < r2ai
  r1aiUR = r2ai
  r1aiLR = r1ai
```

Case 2: r_1a_i range is set and r_2a_i range is not set

```
If r2ai > r1aiUR
  r1aiUR = r2ai
If r2ai < r1aiLR
```

$r_1a_{iLR} = r_2a_i$

Case 3: r_1a_i range is not set and r_2a_i range is set

```
If r1ai > r2aiUR
 r1aiUR = r1ai
 r1aiLR = r2aiLR
If r1ai < r2aiLR
 r1aiUR = r2aiUR
 r1aiLR = r1ai
If r2aiLR <= r1ai <= r2aiUR
 r1aiUR = r2aiUR
 r1aiLR = r2aiLR
```

Case 4: both r_1a_i and r_2a_i ranges are set

```
If r2aiLR < r1aiLR
  r1aiLR = r2aiLR
If r2aiUR > r1aiUR
  r1aiUR = r2aiUR
```

2.3.2 Reasoning at the Lower Level

This section describes the process of reasoning with the instance information collected at the lower level of the structure as described in Section 2.1.1. 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 that resulted from splitting of an original rule. Hence this process happens after a number of iterations where reasoning at the higher level of the structure occurred. The Symmetrical Tau (**τ**) [7] feature selection criterion is used and its calculation is enabled with the instance information collected at the lower level of the structure (Section 2.1.1).

Simplification or rules using the τ criterion. The first step is to calculate the τ measure for each attribute a_i (where i = (1, …, *m*)) in the weight vector $WV(r_p)$ of a rule r_p. Once the τ measure has been calculated for each attribute a_i in $WV(r_p)$ all a_i are ranked according to decreasing τ value. A relevance cut-off is determined in the ranking and it occurs at an attribute if its τ value is less than half of the previous attribute's τ value in the ranking. At this point and below in the ranking all attributes are considered as irrelevant for that rule. On the other hand, if some of the attributes above the relevance cut-off point were previously excluded from $WV(r_p)$, they are now re-introduced since their τ value indicates their relevance for the rule at hand. Once the attribute relevance has been determined for all rules the whole process of rule optimization continues with the main difference being that not all terms in the *ED* formula will be calculated since some attribute constraints do not form the necessary part of the rule any more. When calculating the *ED* between a rule r_p and the input vector IV_e, the i^{th} term of the *ED* formula will be excluded if the attribute a_i is irrelevant. Similarly when calculating the *ED* between two rules r_1 and r_2 the i^{th} term of the *ED* formula will be excluded if the

attribute a_i is irrelevant for both rules. If a_i is irrelevant for one of the rules but not for both, then the rules will not be considered similar to be merged.

3 Method Evaluation

This section describes some experiments performed on a number of real world datasets obtained from the 'uci' machine learning repository [8]. The training dataset was made up of about 70% of randomly chosen instances from the original dataset while the rest was used for testing. Due to space limitations, a detailed comparison with other knowledge learning methods is not provided, but the results are comparable to those obtained by other inductive learners.

Table1: Learned rules from the 'Iris' dataset

Rule 1: 0 < sepal-length < 0.417 AND 0.125 < sepal-width < 1.0 AND 0 < petal-length < 0.153 AND 0 < PW < 0.208 → Iris-setosa
Rule 2: 0.644 < petal-length < 1.0 AND 0.542 < petal-width < 1.0 → Iris-virginica
Rule 3: 0.361 < sepal-length < 0.472 AND 0.417 < sepal-width < 0.583 AND 0.593 < petal-length < 0.644 AND 0.583 < petal-width < 0.708 → Iris-versicolor
Rule 4: 0 < sepal-width < 0.542 AND 0.339 < petal-length < 0.695 AND 0.375 < petal-width < 0.667 → Iris-versicolor

Table 2: Learned rules from the 'Wine' dataset

Rule 1: 0.0 < Alcohol < 0.74 AND 0.03 < Malic_acid < 1.0 AND 0.18 < Ash < 1.0 AND 0.09 < Magnesium < 0.53 AND 0.04 < Total_phenols < 0.88 AND 0.14 < Flavanoids < 1.0 AND 0.05 < Color_intensity < 0.4 AND 0.2 < Hue < 1.0 AND 0.2 < OD280/OD315_diluted_wines < 0.89 AND 0.0 < Proline < 0.43 → Two
Rule 2: 0.48 < Alcohol < 1.0 AND 0.12 < Malic_acid < 0.65 AND 0.36 < Ash < 0.99 AND 0.03 < Alcalinity_of_ash < 0.74 AND 0.21 < Magnesium < 0.67 AND 0.42 < Total_phenols < 1.0 AND 0.39 < Flavanoids < 0.76 AND 0.08 < Nonflavanoid_phenols < 0.7 AND 0.26 < Proanthocyanins < 0.8 AND 0.19 < Color_intensity < 0.65 AND 0.28 < Hue < 0.65 AND 0.45 < OD280/OD315_diluted_wines < 1.0 AND 0.29 < Proline < 1.0 → One
Rule 3: 0.31 < Alcohol < 0.87 AND 0.1 < Malic_acid < 0.97 AND 0.4 < Ash < 0.8 AND 0.36 < Alcalinity_of_ash < 0.85 AND 0.11 < Magnesium < 0.58 AND 0.0 < Total_phenols < 0.63 AND 0.0 < Flavanoids < 0.26 AND 0.08 < Nonflavanoid_phenols < 0.94 AND 0.04 < Proanthocyanins < 0.72 AND 0.22 < Color_intensity < 1.0 AND 0.0 < Hue < 0.39 0.0 < OD280/OD315_diluted_wines < 0.44 AND 0.1 < Proline < 0.43 → Three
Rule 4: 0.1 < Alcohol < 0.52 AND 0.0 < Malic_acid < 0.59 AND 0.0 < Ash < 0.74 AND 0.16 < Total_phenols < 0.8 AND 0.05 < Flavanoids < 0.59 AND 0.02 < Nonflavanoid_phenols < 0.94 AND 0.0 < Color_intensity < 0.38 AND 0.17 < Hue < 0.79 0.12 < OD280/OD315_diluted_wines < 0.82 AND 0.03 < Proline < 0.5 → Two

The learning parameters for *Iris* dataset were set as follows: *iteration#* = 100, *MR* was progressively increased from 0.02 to 0.1 whereas *IR* from 0.00001 to 0.05. The merge value threshold used for merging the value objects of an attribute (see Section 2.1.1) was set to 0.02. The learned rules are displayed in Table 1. Overall the rule set had 93.81% classification accuracy and 96% prediction accuracy. For the *Wine* dataset the following learning parameters were used:

iteration# = 60, *MR* was progressively increased from 0.05 to 0.5 whereas *IR* from 0.01 to 0.05. The merge value threshold was set to 0.04. A total of 4 rules were obtained which are displayed in Table 2 with the classification accuracy of 98.4% and the predictive accuracy of 98.1%.

4 Conclusion

This paper presented a new knowledge learning method for continuous domains that combines the idea of competitive learning, symbolic rule reasoning and statistics. The main difference with the traditional competitive learning as used in the Self-Organizing Map is that the competition occurs among symbolic rules rather than network units. Hence one useful property of the method is that symbolic rules are available at any time during the learning process. The integration of the statistical feature selection criterion has proven useful for attribute relevance analysis during learning and simplification of the learned rules. Evaluation of the method on real world dataset has demonstrated the effectiveness of the method for extraction of optimal symbolic rules. As part of the future work this method will be evaluated on more complex real world datasets and compared more closely with some of existing rule based systems.

References

1. Sestito, S. and Dillon, S.T.: *Automated Knowledge Acquisition*. Prentice Hall of Australia Pty Ltd, Sydney, (1994).
2. Anderson, J.R.: *The Architecture of Cognition*, Harvard University Press, Cambridge, Massachussetts, (1983).
3. Chandrasekaran, B., Goel, A. and Allemang, D.: Connectionism and information processing abstractions. In *AI Magazine*, vol.9, no. 4, Winter, pp. 25-34, American Association for Artificial Intelligence, (1988).
4. Hadzic, F. & Dillon, T.S.: CSOM: Self Organizing Map for Continuous Data, In *Proceedings of the 3rd International IEEE Conference on Industrial Informatics (INDIN'05)*, 10-12 August, Perth, (2005).
5. Kohonen, T.: The Self-Organizing Map. In *Proceedings of the IEEE*, Vol. 78, no 9, pp. 1464-1480, September, (1990).
6. Hadzic, F. and Dillon, T.S.: Rule Optimizing Technique Motivated by Human Concept Formation, In *Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing (BIOSIGNALS 2008)*, January 28-31, Funchal, Madeira, Portugal, (2008).
7. Zhou, X.-J.M. and Dillon, T.S.: A statistical-heuristic feature selection criterion for decision tree induction. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8), 8, 1991, pp. 834-841, (1991).
8. Blake, C., Keogh, E. and 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].