CRNN: Integrating Classification Rule and Neural Network

Wei Li, Longbing Cao and Dazhe Zhao

Abstract—Association classification has been an important type of the rule based classification. A variety of approaches have been proposed to build a classifier based on classification rules. In the prediction stage of the extant approaches, most of the existing association classifiers use the ensemble quality measurement of each rule in a subset rules to predict the class label of the new data. This method still suffers the following two problems. The classification rules are used individually that the coupling relations between rules are ignored in the prediction. However, in real-word rules set, the rules are often inter related together and many rules are usually satisfied partially when a new data object comes. Furthermore, the classification rule based prediction model lacks a general expression of the decision methodology. This paper proposes a classification method that integrating classification rule and neural network (CRNN for short), which gives a general form of the rule based decision methodology by rules network. In comparison with the extant rule based classifiers, such as C4.5, CBA, CMAR and CPAR, our approach has two advantages. First, CRNN takes into account of the coupling relations between rules from the training data in the prediction step. Second, CRNN obtains higher performance on the structure and parameter learning automatically than traditional neural network. CRNN uses the linear computing algorithm in neural network instead of the costly iterative learning algorithm. Two ways of the classification rule set generation are conducted in this paper for the CRNN evaluation and CRNN achieves the satisfied performance.

I. INTRODUCTION

In recent years, the association rule mining integrated with classification, which is called association classification, has been widely studied [1][3]. The performance of association classification, such as CBA [2][4] and CMAR [5], shows that it is even better than traditional rule based classifier such as C4.5 [6]. These methods use association rule mining algorithm, such as Apriori [7], FP-growth [8], to generate a lot of rules and adopt strategies to select useful rules for the classification tasks. The general prediction schema of the rule based classification is shown as in Figure 1. Three methods are usually used in the prediction task, which are single rule based prediction, top-K rules based prediction and group rules together based prediction.

As shown in Fig. 1, the single rule based prediction relies on the sorting of the rules, and the first satisfied rules class is thought as the predicting result. It is more reliable using top-K satisfied rules to make the final decision in the second approach. The measurement (e.g. expected accuracy is used in [14]) of each rule is conducted, and then makes an average value on the K rules that belong to same class. Accordingly, the class with the highest value is selected. However, the rules are not always consistent in class tags for a new coming data object. Therefore, the rules are grouped by the class labels, and then the classifier uses the overall effects of the group rules and yields to the group with highest total measurement, which is as shown in the third method. E.g. $\chi^2$ is used to qualify every rule in the group [5], and then a total sum (or weighted sum) value is obtained. Finally, the label of the group with the highest $\chi^2$ value is chosen as the predicting outcome.

However, the methods shown in Fig. 1 may suffer some weakness as shown below.

1) The coupling relations between the rules are ignored in prediction stage. The rules are treated individually facing the new data object. In fact, the rules bodies often contain same items to each other. The new data object may be fully or partly satisfy to many rules, and these rules have similar rule bodies that means the rules have inter relation and synthesize effect on the class prediction. Moreover, a subset rules are used in the prediction decision as shown in Fig. 1 in extant approaches.

2) A general expression of the rule based decision methodology is lacked at present. The rule’s support and confidence, which are originally used to measure the rules, are abandoned in these methods but using some alternative measurements which have the similar functions as support and confidence theoretically. For example, the Laplace accuracy of a rule $r$ is defined as $(n_r + 1)/(n_{tot} + k)$, where $k$ is the number of classes, $n_{tot}$ is the total number of data objects that satisfy the rule body $p$, and $n_r$ data objects belong to class $c$. However, the value of Laplace accuracy is just the
The framework of the CRNN is shown as Fig. 2. The approach is divided into two parts. Firstly, two ways are shown to generate the classification rules sets. CRNN model is design theoretically and then the construction algorithm is introduced. Secondly, the classification procedure of the CRNN is described using the above model. 21 data sets are used to evaluate the classifier and the approach achieves satisfied performance.
III. CLASSIFICATION RULE GENERATION

In this paper, two different ways are used to generate the rules set, that are the association rule mining (ARM for short) based rule generation and the FOIL based rule generation. A small data set is shown in Example 1.

Example 1 (Mining Classification Rules) Let D is the training data set as Table I (the first 6 columns). There are four attributes in every record and three class labels in total.

<table>
<thead>
<tr>
<th>Table I</th>
<th>DATA SET FOR EXAMPLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule ID</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>a2</td>
</tr>
<tr>
<td>2</td>
<td>a3</td>
</tr>
<tr>
<td>3</td>
<td>a3</td>
</tr>
<tr>
<td>4</td>
<td>a1</td>
</tr>
<tr>
<td>5</td>
<td>a1</td>
</tr>
<tr>
<td>6</td>
<td>a2</td>
</tr>
<tr>
<td>7</td>
<td>a1</td>
</tr>
</tbody>
</table>

A. ARM Based Rule Generation

A new data set D’ is organized based on the original data set D, in which each record is converted to a transaction data record with the class label Y. The new data set is shown as Table I (the last column). This new data set is easy to be processed by the traditional association rule mining methods. The process of ARM based rule generation is different from traditional association rule mining that the rule needs consist of a class tag Yi as right part of the rule. The ARM based classification rule generation is defined as in Algorithm 1.

**Algorithm 1:** ARM based rule generation. ARM-ng  
Data: A transaction database, D; the minimum support threshold, σ.  
Result: A rule set, RuleSet.

1. begin  
2. initial RS = ∅  
3. find all the frequent pattern set, pts, that meet support σ  
4. for each p in pts do  
5. [given pattern p = (A ∧ Y, sup) or p = (A, sup) if p does not contain class label Y, then]  
6. [add p in to patternset, patternset[A] = sup]  
7. for each p in pts do  
8. [given pattern p = (A ∧ Y, sup) or p = (A, sup) if p contains class label Y, then]  
9. [add tuple (A, Y, sup, sup/patternset[A] ) into RuleSet]  
10. end

The classification rules generated by Algorithm 1 can be the raw rules set as the input of CRNN model construction. However, the number of the classification rules is usually very large. Therefore, it need to select the high quality rules for classification. There are many selection strategies in the previous works [5][12][13]. In this paper, the database coverage method which is proposed in [2] is used to select the classification rules.

Given two rules R1 and R2, R1 is said having higher rank than R2, if and only if (1) \( \text{conf}(R1) > \text{conf}(R2) \); (2) if \( \text{conf}(R1) = \text{conf}(R2) \), but \( \text{sup}(R1) > \text{sup}(R2) \); (3) if \( \text{conf}(R1) = \text{conf}(R2) \) and \( \text{sup}(R1) = \text{sup}(R2) \), but \( R1 \) has less items than \( R2 \). According to this rules ranking methodology, the classification rules are sorted in descending order firstly. Every rule is tested for how many records are covered in the data set. The rules that cover at least one record are selected until all the data records are covered with a user predefined minimum threshold.

The new rule set with the rules selection strategy is much smaller than the original one. If the minimum data set coverage is set to 2 in Example 1, only 3 rules in Table II are remained, that are \( r_{01} : a_2, c_2 \rightarrow Y_1 \), \( r_{02} : a_1, d_3 \rightarrow Y_3 \) and \( r_{09} : a_1 \rightarrow Y_2 \).

B. FOIL Based Rule Generation

FOIL is used to find the rules that distinguish the positive examples from the negative ones [11]. Usually, FOIL is applied on each class when the data set has multiple class labels. The multiple class problems are transformed into several binary class problems. Rules are obtained for every class and then the rules for each class are merged to form the final rule set of the whole data set. The rules generated by the FOIL do not fit with the CRNN model since the rules lack the support and confidence parameters. The support and confidence need to be appended additionally.

In FOIL based rule generation procedure, a measurement is required to constrain how to select an attribute value to form a rule. The FOIL gain is usually used to get the information gain when an attribute value, \( A_i \), is added to the current rule \( r \). Let \( |P| \) be the number of the positive
examples and \(|N|\) is the number of the negative examples. Once an attribute value, \(A_i\), is added to the \(r\)'s body, we get the new numbers of the positive and negative examples, as \(|P'\) and \(|N'|\). Thus the FOIL gain of \(A_i\) is computed as following.

\[
fgain(A_i) = |P'| \left( \log \frac{|P'|}{|P'| + |N'|} - \log \frac{|P|}{|P| + |N|} \right)
\] (3)

The attribute value with maximum FOIL Gain is selected to append the rule body until all the positive examples are covered. FOIL based rule generation is presented in Algorithm 2.

**Algorithm 2: FOIL based rule generation.**

Data: A data set, \(ds\); the minimum foil gain threshold, \(\delta\).
Result: A rule set, \(RuleSet\).

1 begin
2 \[initial \ RuleSet = \emptyset,\] given class labels set \(Y = Y_1, \ldots, Y_M\) for each \(c\) in \(Y\) do
3 \[initial \ rs4c = \emptyset \] for class \(c\)
4 get positive and negative data sets, \(PD, ND\) of \(c\)
5 while \(PD \neq \emptyset\) do
6 \[rbody = \emptyset, PD' = PD, ND' = ND\]
7 while \(|ND'| > 0 \) and \(rbody.length < \max - rule - len\) do
8 searching \(A_i\), with maximum \(fgain\)
9 if \(fgain(A_i) < \delta\) then
10 \[break\]
11 adding \(A_i\) into the \(rbody\)
12 delete examples not satisfied \(rbody\) in \(PD', ND'\)
13 adding the \(rbody\) into \(rs4c\)
14 delete all the examples satisfied \(rbody\) in \(PD\)
15 for each \(rb\) in \(rs4c\) do
16 computing the \(sup(r)\) and \(conf(r)\) of rule \(r : rb \rightarrow c\)
17 adding rule tuple \((rb, c, sup(r), conf(r))\) into RuleSet
18 end

Four rules are generated from the data set shown in Example 1 where the \(fgain\) is set to 0.5. The rules set are \(\{a_2 \rightarrow Y_1, b_3, c_3 \rightarrow Y_1, c_1 \rightarrow Y_2, d_3, a_1 \rightarrow Y_3\}\).

**IV. CRNN MODELLING**

According to the ANN classification, we define the CRNN model structure. The CRNN structure relies on the rules set found in data set.

**A. CRNN Model Design**

The multilayer ANN structure is common seen in many studies. User usually needs to determine the input and output nodes depended on the problem and the number of nodes in the hidden layer as well. Thus a parameter learning algorithm is used to learn the weight parameters between the nodes in ANN. Similarly, the CRNN model is designed as a four layers neural network which includes the input layer (IN layer), pattern and class middle layers (PN and CN layer) and output layer (ON layer). We call the hidden layer as a middle layer for that the nodes in the middle layers of CRNN can be interpreted as actual meaning. The structure of CRNN is shown in Fig. 3.

In comparison with traditional ANN, There are some differences in the CRNN model.

- The nodes between the layers are partially connected while the nodes in ANN are not. In particular, PN node has and only has an input link, and CN node has and only has an output link.
- The nodes in the middle layers of CRNN stand actual meaning. Furthermore, the nodes and links between the middle layers are determined by classification rules set obtained from the training data set. Different dataset falls into different CRNN model structures automatically.
- The parameters, \(X_q, w_{qi}, w_{ij}, w_{jk}\) and \(Y_k\), in CRNN require no complex and time-consuming learning algorithms (such as back propagation) to be computed out. These parameters are all obtained from the classification rules set.

**B. CRNN Construction Using Rule Set**

Every classification rule actually contains five data elements: the rule id, attribute items, class label, support and confidence, which is written as: \(\{RID : (set_of_items, class, sup, conf)\}\). The mapping between the elements of a rule and the nodes of the four layers CRNN structure are shown as Fig. 4.

The detail mapping steps are described as following.

- The input nodes in CRNN stand for the attributes of the data record. That means the dimension of data set is the number of the input nodes. For example, the supermarket retail data can be presented by only one node in CRNN. The data set in Example 1 has four attributes, \(A, B, C, D\), thus four input nodes appear in the CRNN.
The nodes in the first middle layer contain all the items that appear in the rule body. Every input node has links to its own attribute value.

The nodes in the second middle layer correspond to the classification rules. Every rule is represented as one node in the CN layer. The connections between the PN layer and CN layer depend on whether the items in PN and CN layers are contained in a same rules.

The output nodes represent the class labels of the data set, and every node in the CN layer only links to its class node in the ON layer.

\( w_{jk} \) and \( w_{ij} \) are the support and confidence of the rule respectively. \( w_{ij} \) in CRNN gives the confidence of the rule in CN when the items in the rule body appear in PN node. \( w_{jk} \) represents the linkage of rule \( CN_j \) and class \( Y_k \).

\( w_{qi} \) is always 1.0 which means the attribute \( X_q \) has items of attribute value in \( PN_i \).

Based on the above steps of CRNN structure building and parameter setting, the CRNN model construction is presented in Algorithm 3.

Given a classification rule set, \( RuleSet = \{ r_1, r_2, \ldots, r_R \} \), every rule \( r \) has \( n_r \) items in the rule body, the class label \( Y_r \) in \( \{ Y_1, Y_2, \ldots, Y_M \} \). The rule \( r \) in RuleSet is formed as \( (RuleID_r, P_r, Y_r, sup_r, conf_r) \). The CRNN is represented by a graph data structure \( G = (V, E) \).

**Example 2 (CRNN Construction)** 14 rules in Example 1 are used to construct a CRNN model by Algorithm 3.

The first rule is \( r_{01} : a_1 \rightarrow Y_3 \). The nodes, \( X_A, PN_{A_1}, CN_{r_{01}} \) and \( ON_{Y_3} \), the linkages \( (X_A, PN_{A_1}, 1.0), (PN_{A_1}, CN_{r_{01}}, 0.41) \) and \( (CN_{r_{01}}, ON_{Y_3}, 0.29) \) are created. For the rule \( r_{02} : a_1, d_3 \rightarrow Y_3 \), the node \( X_A \) and \( ON_{Y_3} \) have been exists, thus only the nodes, \( X_D, PN_{D_3} \) and \( CN_{r_{02}} \) need to be appended into the network. The edge between \( X_A \) and \( PN_{A_1} \) has been already in the CRNN. The links \( (X_D, PN_{D_3}, 1.0), (PN_{D_3}, CN_{r_{02}}, 1.00), (PN_{D_3}, CN_{r_{02}}, 1.00) \) and \( (CN_{r_{02}}, ON_{Y_3}, 0.29) \) are added. All the nodes and edges eventually are obtained in CRNN until all the rules have been processed. We will get to the final neural network structure as shown in Fig. 5.

The construction of the CRNN takes \(|RQ|\) time in CRN-NCON algorithm relying on the rules set size and rule body length. The \( R \) is the number of rule set and \( Q \) is the length of items in rule set, that \( Q = n_1 + n_2 + \cdots + n_R \). Through the above description, we get the following characteristics of the CRNN classification model.

- **Structural autonomy**: CRNN connections between the nodes are determined by association rules, and IN nodes and PN nodes are fixed by the data itself naturally. Input layer of CRNN has the same number of the nodes as the dimension of the data set attributes. The PN nodes in the first middle layer are related to the items in the association rules. There are identical CN nodes to the
rules. CRNN network can be fixed without user manual design contrasting with the traditional multi-layer neural network.

- **Parameter setting**: The parameters in the CRNN network are assigned directly. The weights between the IN nodes and PN nodes are Boolean type. If the item of the PN node, which belongs to a attribute, appears in an association rule, then the weight between the input node and the PN node is set to 1, otherwise it is set to 0 (the links in current building process and examples are not considered). The weights, \(w_{ij}\) and \(w_{jk}\), are also from the association rules. CRNN parameters are not obtained by learning algorithms as the traditional neural network, which makes the CRNN construction quickly.

- **Interpretable relationship**: The weights \(w_{ij}\) describe the relations between patterns and classes. This is identical to the meaning of the classification rule, that when the pattern appears, the class label will be predicted with confidence. The relations between the CN and ON nodes are obvious that the class labels are predicted correctly with the probabilities. The supports of the rules just represent the frequency in the data set (approximate with probability).

- **Transparent layers**: As we know, the hidden layer in the traditional neural network is design by user and the meaning of the hidden node is usually hard to be interpreted for user. But this situation is changed in the CRNN. The hidden layer which is called middle layer in the CRNN has intuitive explanation and is transparent to the user.

V. PREDICTION USING CRNN

This section introduces a CRNN classifier for the new data object prediction.

A. Building a Classifier

There are many activation functions used in the traditional ANN, such as threshold function, piecewise linear function, sigmoid function, Gaussian function and so on. A piecewise linear function is used in this paper, which is defined as \(F : y = x, x \in (0, 1)\), as shown in Fig. 6(a). The parameters of CRNN can not be negative, and the maximum value of \(x\) is 1.0.

![Fig. 6. CRNN activation function and node computing](image)

General speaking, the node computing model in the neural network is shown as in Fig. 6(b). Given a neural node has \(n\) input links, \(X = (x_1, x_2, \ldots, x_n)^T\), and the weights for every link are represented as \(W = (w_1, w_2, \ldots, w_n)^T\). The output \(Y\) is computed as following.

\[
Y = F(W^T X + b) = F \left( \sum_{i=1}^{n} w_i x_i + b \right) = \sum_{i=1}^{n} w_i x_i + b
\]

(4)

where the \(b\) is the bias and \(F\) is the activation function.

Let the rules set be \(\text{RuleSet} = \{r_1, r_2, \ldots, r_R\}\), the class labels set \(\{Y_1, Y_2, \ldots, Y_M\}\), and each class \(Y_k\) has \(n_k\) rules, that are \(r_{1k}, r_{2k}, \ldots, r_{nk}\) \((1 \leq k \leq M)\), and \(R = \sum_{k=1}^{M} n_k\). Every rule \(r_{pk}\) has the support as \(s_{pk}\), the confidence as \(c_{pk}\). The items in the rule body are assumed as \(\{I_1^p(Y_k), I_2^p(Y_k), \ldots, I_{nk}^p(Y_k)\}\). Then the output node \(Y_k\) can be computed according to the above node computing model.

\[
Y_k = \sum_{t=1}^{n_k} F(r_t) w_t + b_k^e
\]

(5)

\[
= s_{1k}^r F(r_1) + s_{2k}^r F(r_2) + \cdots + s_{nk}^r F(r_{nk}) + b_k^e
\]

\(F(r_t)\) is the output of the node in the CN layer, and \(F(r_t)\) is obtained as following.

\[
F(r_t) = \text{conf}(I_1^t(Y_k)) F(I_1^t(Y_k)) + \cdots + \text{conf}(I_{nk}^t(Y_k)) F(I_{nk}^t(Y_k))
\]

(6)

\[
= c_{1k}^r \psi(I_1^t) + c_{2k}^r \psi(I_2^t) + \cdots + c_{nk}^r \psi(I_{nk}^t) + b_k^e
\]

Since the weights between the IN layer and PN layer are Boolean values, \(\psi\) is indicator function that \(\psi = 1\) when PN node has output, otherwise \(\psi = 0\). Assuming a Boolean vector, \(V_t = (1, 1, \ldots, 1, 0, \ldots, 0)\), is the input vector of the CN layer where the element with zero value stands for there is no link to CN node. Thus the value of \(r_t\) is obtained as:

\[
F(r_t) = (c_{1k}^r, c_{2k}^r, \ldots, c_{nk}^r, 0, \ldots, 0) V_t^T + b_t^r
\]

(7)

We take this value into the \(Y_k\), then we will get the \(Y_k\).

\[
Y_k = s_{1k}^r \psi(I_1^t) + \cdots + s_{nk}^r \psi(I_{nk}^t) + b_k^e = (s_{1k}^r, s_{2k}^r, \ldots, s_{nk}^r) V_t^T + b_k^e
\]

(8)
$V_1, V_2, \cdots, V_{n_k}$ is normalized by unify order. E.g. one sorts the rules by the rules’ number. A new Boolean vector $U$ from $V$ is got according to the rules order. The final output value of class, $Y_k$, is represented as following.

$$Y_k = \left( s_1 Y_k, s_2 Y_k, \ldots, s_{n_k} Y_k \right) \begin{pmatrix} U_1 I \\ U_2 I \\ \vdots \\ U_{n_k} I \end{pmatrix}$$

$$+ \left( s_1 Y_k, s_2 Y_k, \ldots, s_{n_k} Y_k \right) \begin{pmatrix} b_1^c \\ b_2^c \\ \vdots \\ b_{n_k}^c \end{pmatrix}$$

(9)

where $I$ is a unit column vector, namely $I^T = (1, 1, \cdots, 1)$. According to the structure of the CRNN, $U_k I$ is just the number of the items in the rule $r_k$. When the new data comes, the above method is used to compute $Y$ value for every class. The class label with the maximum value of $Y$ is selected as the class label of the new data. However, one problem is still in the formula (9) because the parameters of $b$ are unknown. In this paper, we set all the parameter of $b$ to zero. The final computing model of $Y_k$ is shown as following formula.

$$Y_k = \left( s_1 Y_k, s_2 Y_k, \ldots, s_{n_k} Y_k \right) \begin{pmatrix} U_1 I \\ U_2 I \\ \vdots \\ U_{n_k} I \end{pmatrix}$$

(10)

All the $s$ and $c$ are known for us and the $U$ can be known if user gives the new data. The class $k$ with maximum of $Y_k$ is regarded as the class label of the new data, which is shown as formula 11.

$$k_{predict} = \arg\max\{Y_k\}, k \in \{1, 2, \cdots, M\}$$

(11)

Example 3 (CRNN Classification) Let the new data be $x = (A = a_1, B = b_2, C = c_2, D = d_1)$, predicting the class label of $x$ using the CRNN model as shown in Fig. 5. All the values of $Y$ are calculated as following.

$$Y_1 = \left( s_8 Y_1, s_9 Y_1, s_{12} Y_1, s_{13} Y_1, s_{14} Y_1 \right) \begin{pmatrix} U_8 I \\ U_{12} I \\ U_{13} I \\ U_{14} I \end{pmatrix}$$

(12)

The computing processes of $Y_2$ and $Y_3$ are same as $Y_1$. The $Y_2$ and $Y_3$ are 0.264 and 1.085 relatively. The maximum value is $Y_3$, so the class of the new data $x$ is $Y_3$.

B. General Form of Rule-based Decision Methodology

It is known that the $w_{ij}$ is $conf(r)$ and $w_{jk}$ is $sup(r)$ in Algorithm 3. The value of $Y_k$ is transformed as following when the $w_{ij}$ is set to be $conf(r)/L$, where $L$ is the length of the rule body.

$$Y_k = \left( s_1 Y_k, s_2 Y_k, \ldots, s_{n_k} Y_k \right) \begin{pmatrix} U_1 I \\ U_2 I \\ \vdots \\ U_{n_k} I \end{pmatrix}$$

(13)

where $U_k I$ is just the number of the items in the rule $r_k$. This is a kind of general form for the rule based decision process. Let us exam the three methods in Fig. 1, and we can get that they are the special forms of the above decision expression where the node activation function $F$ is set as threshold type. We assume the CN node is in activation when all the PN nodes (items of rule body) have output, which means the rule is fully satisfied as in [4][5][14].

- We treat the $Y_k$ (sum of $s_i Y_k c_i Y_k$) in another way, not in sum expression. All $s_i Y_k c_i Y_k$ within $Y_k$ are sorted in descending. The class label with first highest value $s_i Y_k c_i Y_k$ is selected which is the first method described as in [4].
- For the second method in Fig. 1, the satisfied top-$K$ rules are selected for each class $Y_k$, and the class label with highest $Y_k$ value of the $K$ rules (i.e. $K = 5$) is treated as predicting class. For example, the Laplace accuracy is similar with the confidence when the data set size is big enough. The class label with highest arithmetic average of Laplace accuracy of the top-$K$ rules is regarded as the predicting tag in [14]. The $Y_k$ here is the weighted average of confidence (the supports in $Y_k$ is the weights sum of confidences, because the sum of the supports for $Y_k$ is 1.0 in Algorithm 3 [line 15]).
- Let the rule be $r : p \rightarrow c$, the sum of $s_i Y_k c_i Y_k$ (sup$(r)$conf$(r)$ for short) is equal as following: $Y_k = \sum sup(r)conf(r) = \sum sup(p,c)sup(p,c)/sup(p)$ The above $Y_k$ has similar function with the sum of $\chi_i^2$ for each rule in the group [5].

VI. EXPERIMENTAL RESULTS

In this paper, 21 data sets from UCI ML Repository are used to evaluate our approach. Discretization of continuous attributes is done using the same method in [15]. We have conducted the accuracy study on these data and compared CRNN with C4.5 [6] and CPAR [14]. The CRNN algorithm is implemented in Python v2.7, and all the other approaches are tested by their authors. All experiments are conducted on a desktop computer with Intel Core 2 CPU of 2.80GHz, 4GB memory and Windows 7. The rules sets are generated by two different ways, which are ARM based rule generation (CRNN-a for short) and FOIL based rule generation (CRNN-f for short). The performance of CRNN on the two rule sets is shown as in Table III.

The parameters of the CRNN model are set as following. The support and confidence of association rule mining are set to 0.05 and 0.9. The database coverage threshold is set to 10. With regard to the FOIL based rule generation, the fgain
and rules for the CPAR, CRNN-a and CRNN-f on the 21 datasets. We compute the arithmetic and geometric average of the running time and average number of rules used in CRNN. We compute the average accuracy of CRNN as bold font in Table III.

As the result shown in Table 4, the average accuracy of CRNN is higher than C4.5. CRNN-a and CRNN-f have higher performance in some data set which are shown in bold font in Table III.

**Table IV**

<table>
<thead>
<tr>
<th>Average</th>
<th>CPAR</th>
<th>CRNN-c</th>
<th>CRNN-f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Rules</td>
<td>Time</td>
<td>Rules</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>0.2</td>
<td>86.5</td>
<td>85.9</td>
</tr>
<tr>
<td>Geometric</td>
<td>0.3</td>
<td>74.4</td>
<td>65.5</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>71.0</td>
<td>72.0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>84.4</td>
<td>84.3</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>95.2</td>
<td>95.1</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>11.4</td>
<td>11.7</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>14.4</td>
<td>14.7</td>
</tr>
</tbody>
</table>

Table IV shows the algorithm running efficiency (in sec.) and average number of rules used in CRNN. We compute the arithmetic and geometric average of the running time and rules for the CPAR, CRNN-a and CRNN-f on the 21 datasets.

**References**