CRNN: Integrating Classification Rules into Neural Network

Wei Li, Longbing Cao, Dazhe Zhao, Xia Cui and Jinzhu Yang

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 of rules to predict the class label of the new data. This method still suffers the following two problems. The classification rules are used individually thus the coupling relations between rules [1] are ignored in the prediction. However, in real-world rule set, rules are often inter-related and a new data object may partially satisfy many rules. Furthermore, the classification rule based prediction model lacks a general expression of the decision methodology. This paper proposes a classification method that integrating classification rules into neural network (CRNN, for short), which presents a general form of the rule based decision methodology by rule-based 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 the coupling relations between rules from the training data into account in the prediction step. Second, CRNN automatically obtains higher performance on the structure and parameter learning 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 satisfactory performance.

I. INTRODUCTION

In recent years, the association rule mining integrated with classification, which is called association classification, has been widely studied [2][4]. The performance of association classification, such as CBA [3][5] and CMAR [6], shows that it is even better than traditional rule based classifier such as C4.5 [7]. These methods use association rule mining algorithm, such as Apriori [8], FP-growth [9], to generate a lot of rules and adopt strategies to select useful rules for the classification tasks. The general prediction schema of classification rules is shown as in Fig. 1. Three methods are usually used in the prediction task, which are single rule based prediction, top-K rules based prediction, and group rules based prediction.

As shown in Fig. 1, the single rule based prediction relies on the sorting of the rules, and the class label of the first satisfied rule is taken as the predictive 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 [15]) of each rule is conducted, and then an average value on the K rules that belong to same class will be generated. 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 performance, which is as shown in the third method. E.g., \( \chi^2 \) is used to measure the quality of rules in the group [6], and then the total sum (or weighted sum) value is obtained. Finally, the label of the group with the highest \( \chi^2 \) value is chosen as the prediction outcome.

Fig. 1. The classification rule based prediction methodology

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 in determining the new data object. In fact, different rules often contain same items. The new data object may fully or partly satisfy many rules, and these rules have similar or partially similar rule logic that means these rules share inter-relation and as a whole contribute to the class prediction. Moreover, a subset of 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 while some alternative measurements which have the theoretically similar functions as support and confidence are used for. For example, the Laplace accuracy of a rule \( r \) is defined as \( \frac{n_c + 1}{n_{tot} + k} \), where \( k \) is the number...
of classes, \( n_{\text{tot}} \) is the total number of data objects that satisfy the rule body \( p \), and \( n_c \) data objects belong to class \( c \). However, the value of Laplace accuracy is just the confidence of rule when the data set size \( N \) is big enough.

**Proof:** \( (n_c + 1)/(n_{\text{tot}} + k) = (n_c/N + 1/N)/(n_{\text{tot}}/N + k/N) \approx (n_c/N)/(n_{\text{tot}}/N) = \text{support}(r)/\text{support}(p) = \text{confidence}(r) \), where \( 1/N \approx 0 \) and \( k/N \approx 0 \).

In order to solve the above two problems, rules are used in prediction model as rule network. The support and confidence are also integrated into the classification model. The idea of this paper is from artificial neural network [11] (ANN for short) which can cover all the rules and rules parameters. As we known, the ANN classifier has been studied in machine learning for many years and it has been used in a variety of applications. ANN has many advantages in practice, but a coin has two sides. The training process is usually longer than other classification approaches; moreover, the structure of ANN needs to be designed manually and the hidden nodes form a black box which is hard for user understanding.

This paper proposes a new neural network structure based on the classification rules (CRNN for short), which integrates the advantages of association classification and neural networks and is different from [10]. This new classification method makes full use of classification rules and obtains much more efficiency in both the network structure and parameter learning than the traditional neural networks. In our approach, CRNN relies on the rules mined from the training data set, so the quality of rules has still impact on the performance of CRNN. We use two different rule set generation methods, which are the association rule mining with rule selection strategy [6][9] and FOIL (First Order Inductive Learner) based rule searching with the rule support and confidence [12], to conduct the CRNN prediction accuracy evaluation. CRNN tackles the problem of the general expression of rule based prediction decision and complex computing process of the structure and parameter learning in neural network. The contributions of this study are as follows.

1) We propose a method to integrate all the classification rules into classification model as well as the coupling relations between rules.
2) This paper introduces a new structure of neural network. CRNN is created by the rule set and the structure of CRNN is determined automatically.
3) CRNN shows a new approach to obtain the parameters in neural network efficiently. The parameters in CRNN are fast obtained once the classification rules are given.

The subsequent sections are organized as follows. Section II describes the definition of the research problem and an overview of the proposed method. Section III introduces the two methods of classification rule generation. Section IV shows the design and construction of the CRNN model. CRNN classification is presented in Section V using the CRNN model. Our approach is evaluated in Section VI and we make the conclusions in Section VII.

**II. Problem Statement**

Given a set of data items \( I = I_1, I_2, \ldots, I_n \), where \( I_1, I_2, \ldots, I_n \) are items of attributes values. Let \( D \) denote the data set which has \( N \) data records. Each record contains a number of attributes and every attribute contains a subset items in \( I \). Each continuous attribute should be firstly discretized into categorical attributes items. Each data record corresponds to a class tag of \( Y \), where \( Y = \{Y_1, \ldots, Y_M\} \).

**Definition 1: (Classification Rule)** Let pattern \( A \) contain one or more data items combined as \( I_1^c \land \ldots \land I_j^c \), where \( A \) links to a class tag \( Y_k \), we call the form, \( r : I_1^c \land \ldots \land I_j^c \rightarrow Y_k \), a **Classification Rule**. The support of the classification rule \( r \) is defined as the percentage of the pattern \( A \) in the data set, written as \( \text{sup}(r) \).

\[ \text{sup}(r) = \frac{\text{sup}(A^c \land Y_k)}{N} = \frac{\#(A^c \land Y_k)}{N} \tag{1} \]

where \( \#(A^c \land Y_k) \) is the size of the data records that cover pattern \( A \) and belong to class \( Y_k \). The confidence of the classification rule, \( \text{conf}(r) \), is defined below.

\[ \text{conf}(r) = \frac{\text{sup}(A^c \land Y_k)}{\text{sup}(A)} = \frac{\text{sup}(r)}{\text{sup}(A)} \tag{2} \]

The problem we want to solve is as follows. Defining a model \( F \) based on neural network that incorporates the classification rule set, \( \{r_1, r_2, \ldots, r_p\} \), data set \( D \), then for the new data record \( x \), making prediction using the model \( F \), namely \( Y_k = F(r_1, r_2, \ldots, r_p)_x \).

**Fig. 2.** CRNN framework
model is theoretically designed 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 satisfactory performance.

### III. Classification Rule Generation

In this paper, two different ways are used to generate the rule 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 $D$ be the training data set as in Table I (the first 6 columns). There are four attributes in every record and three class labels in total.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Class</th>
<th>Transaction record</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₂</td>
<td>d₁</td>
<td>Y₁</td>
<td>a₂b₁c₂d₁Y₁</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
<td>c₁</td>
<td>d₂</td>
<td>Y₂</td>
<td>a₁b₂c₁d₂Y₂</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₂</td>
<td>d₁</td>
<td>Y₁</td>
<td>a₂b₁c₂d₁Y₁</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
<td>c₁</td>
<td>d₁</td>
<td>Y₂</td>
<td>a₁b₂c₁d₁Y₂</td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₂</td>
<td>d₁</td>
<td>Y₂</td>
<td>a₁b₂c₂d₁Y₂</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
<td>c₁</td>
<td>d₂</td>
<td>Y₂</td>
<td>a₁b₂c₁d₂Y₂</td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>d₂</td>
<td>Y₁</td>
<td>a₁b₁c₁d₂Y₁</td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>d₂</td>
<td>Y₁</td>
<td>a₁b₁c₁d₂Y₁</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 $Y_i$. The new data set is shown in 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, in which a rule consists of a class tag $Y_i$ as the right part of the rule. The ARM based classification rule generation is defined as in Algorithm 1.

**Algorithm 1: ARM based rule generation, ARM-r**

- **Data:** A transaction database, $D$; the minimum support threshold, $\sigma$.
- **Result:** A rule set, RuleSet.

1. begin
2. initialize $RS = \emptyset$
3. find all the frequent pattern set $pts$ that meets support $\sigma$
4. initial pattern and support map table, $patternset = \{\}$
5. for each p in pts do
6. given pattern $p = (A \land Y_i, sup)$ or $p = (A, sup)$ if $p$
7. doesn't contain class label $Y_i$ then
8. add p in to $patternset$, $patternset[A] = sup$
9. for each p in $patternset$ do
10. given pattern $p = (A \land Y_i, sup)$ or $p = (A, sup)$ if $p$
11. contains class label $Y_i$ then
12. add tuple $(A, Y_i, sup, sup/patternset[A])$ into RuleSet
13. end
14. end

The Algorithm 1 has twice scans on the frequent patterns set. The first traversal is used for computing the classification rule confidence, while the second traversal finds all the classification rules with support and confidence parameters. 14 classification rules in Example 1 are obtained as shown in Table II ($sup = 2/7$).

**TABLE II Classification rules mined in Example 1**

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>Rules</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₀₁</td>
<td>a₁ → Y₃</td>
<td>0.29</td>
<td>0.44</td>
</tr>
<tr>
<td>r₀₂</td>
<td>a₁, d₃ → Y₃</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₀₃</td>
<td>a₁, b₂ → Y₃</td>
<td>0.29</td>
<td>0.67</td>
</tr>
<tr>
<td>r₀₄</td>
<td>a₁, b₂, d₃ → Y₃</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₀₅</td>
<td>d₃ → Y₃</td>
<td>0.29</td>
<td>0.67</td>
</tr>
<tr>
<td>r₀₆</td>
<td>b₂, d₃ → Y₃</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₀₇</td>
<td>b₂ → Y₃</td>
<td>0.29</td>
<td>0.67</td>
</tr>
<tr>
<td>r₀₈</td>
<td>a₂ → Y₂</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₀₉</td>
<td>a₁ → Y₂</td>
<td>0.29</td>
<td>0.44</td>
</tr>
<tr>
<td>r₁₀</td>
<td>a₁, c₁ → Y₂</td>
<td>0.29</td>
<td>0.67</td>
</tr>
<tr>
<td>r₁₁</td>
<td>c₁ → Y₂</td>
<td>0.29</td>
<td>0.67</td>
</tr>
<tr>
<td>r₁₂</td>
<td>c₂ → Y₁</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₁₃</td>
<td>a₂, c₂ → Y₁</td>
<td>0.29</td>
<td>1.00</td>
</tr>
<tr>
<td>r₁₄</td>
<td>b₃ → Y₁</td>
<td>0.29</td>
<td>0.67</td>
</tr>
</tbody>
</table>

The classification rules generated by Algorithm 1 can be the raw rule set as the input of the CRNN model construction. However, the number of the classification rules is usually very large. Therefore, it needs to select the high quality rules for classification. There are many selection strategies in the previous work [6][13][14]. In this paper, the database coverage method which is proposed in [2] is used to select the classification rules.

Given two rules $R_1$ and $R_2$, $R_1$ has higher rank than $R_2$, if and only if (1) $conf(R_1) > conf(R_2)$; (2) if $conf(R_1) = conf(R_2)$, but $sup(R_1) > sup(R_2)$; (3) if $conf(R_1) = conf(R_2)$ and $sup(R_1) = sup(R_2)$, but $R_1$ has less items than $R_2$. According to this rule ranking methodology, the classification rules are sorted in the 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 generated by the rule 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 $r₁₃: a₂, c₂ → Y₁, r₀₄: a₁, b₂, d₃ → Y₃$ and $r₁₀: a₁, c₁ → Y₂$.

**B. FOIL Based Rule Generation**

FOIL is used to find the rules that distinguish the positive examples from the negative ones [12]. 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 the 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 \( |N| \) be 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 \( |N'| \) respectively. Thus the FOIL Gain of \( A_i \) is computed as follows.

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

(3)

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

**Algorithm 2:** FOIL based rule generation. Foil-rg

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

**Result:** A rule set, \( RuleSet \).

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

Four rules are generated from the data set shown in Example 1 where the \( fgain \) is set to 0.5. The rule 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 rule set found in data set.

A. CRNN Model Design

The multilayer ANN structure is commonly seen. One usually needs to determine the input and output nodes as per the underlying 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 layer neural network which includes the input layer (IN layer), pattern and class layers (PN and CN layer) and output layer (ON layer). We call the hidden layer as a middle layer since that the nodes in the middle layers of CRNN can be interpreted in actual meaning. The structure of CRNN is shown in Fig. 3.

In comparison with traditional ANN, there are some difference 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 for actual meaning. Furthermore, the nodes and links between the middle layers are determined by classification rule set obtained from the training data set. Different datasets fall 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 rule 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: \( \{ RID : \{ \text{set of items, class, sup, conf} \} \} \). The mapping between the elements of a rule and the nodes of the four-layer CRNN structure are shown as Fig. 4.

The detailed mapping steps are described as follows.

- 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 the same rule.

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. \( w_{jk} \) in CRNN gives the confidence of the rule in CN when the items in the rule body appear in the 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 \( Y_r = \{Y_{1r}, Y_{2r}, \ldots, Y_{Mr}\} \). The rule \( r \) in RuleSet is formed as \( (RuleID_r, PN_r, sup_r, conf_r) \). 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, \( X_A, PN_{a_1}, CN_{v_{t1}}, and ON_{Y_3} \), the linkages \( (X_A, PN_{a_1}, 1.0) \), \( (PN_{a_1}, CN_{v_{t1}}, 0.41) \) and \( (CN_{v_{t1}}, ON_{Y_3}, 0.29) \) are created. For the rule \( r_{02}: a_2, d_3 \rightarrow Y_3 \), the nodes \( X_A \) and \( ON_{Y_3} \) exist, thus only the nodes, \( X_D, PN_{d_3} \) and \( CN_{v_{t2}} \) need to be appended into the network. The edge between \( X_A \) and \( PN_{a_1} \) has been already in the CRNN. The links \( (X_D, PN_{d_3}, 1.0) \), \( (PN_{a_1}, CN_{v_{t2}}, 0.41) \), \( (PN_{d_3}, CN_{v_{t2}}, 1.0) \) and \( (CN_{v_{t2}}, ON_{Y_3}, 0.29) \) 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 CRNNCON algorithm relying on the rule set size and rule body length. \( R \) is the number of rule set and \( Q \) is the length of items in rule set, \( Q = n_1 + n_2 + \cdots + n_R \). 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. The 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 by contrast to 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 of Boolean type. If the item of the PN node, which belongs to a attribute, appears in an association rule, thus 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 quick.

- **Interpretable relationship:** The weights $w_{ij}$ describe the relations between patterns and classes. This is identical to the meaning of the classification rule, 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. The relations between patterns and classes. This is identical to the meaning of the classification rule, when the pattern appears, the class label will be predicted with confidence. The relations between patterns and classes. This is identical to the meaning of the classification rule, when the pattern appears, the class label will be predicted with confidence.

- **Transparent layers:** As we know, the hidden layer in the traditional neural network is designed by users and the meaning of the hidden node is usually hard to be interpreted for users. 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 cannot be negative, and the maximum value of $x$ is 1.0.

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

$$Y = F(W^TX + b) = F \left( \sum_{i=1}^{n} w_ix_i + b \right) = \sum_{i=1}^{n} w_ix_i + b$$

where the $b$ is the bias and $F$ is the activation function. Let the rule set be $\text{RuleSet} = \{r_1, r_2, \cdots, r_R\}$, the class labels set $\{Y_{1}, Y_{2}, \cdots, Y_{M}\}$, and each class $Y_k$ has $n_k$ rules, that is $r_{k1}, r_{k2}, \cdots, r_{kn_k}$ ($1 \leq k \leq M$), and $R = \sum_{k=1}^{M} n_k$. Every rule $r_{kp}$ has the support as $s_{kp}$. The items in the rule body are assumed as $\{I_{p1}(Y_k), I_{p2}(Y_k), \cdots, I_{pn_k}(Y_k)\}$. Then the output node $Y_k$ can be computed according to the above node computing model.

$$Y_k = \sum_{i=1}^{n_k} F(r_i)w_i + b_k^c \quad (5)$$

$$= s_{k1}^Y F(r_1) + s_{k2}^Y F(r_2) + \cdots + s_{kn_k}^Y F(r_{kn_k}) + b_k^c$$

$F(r_i)$ is the output of the node in the CN layer, and $F(r_i)$ is obtained below.

$$F(r_i) = \text{conf}(I_{i1}(Y_k))F(I_{i1}(Y_k)) + \cdots + \text{conf}(I_{im}(Y_k))F(I_{im}(Y_k))$$

$$= c_{1}^Y(r_i)S(I_{i1}) + c_{2}^Y(r_i)\psi(I_{i2}) + \cdots + c_{n_k}^Y(r_i)\psi(I_{im}) + b_k^c \quad (6)$$

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

$$F(r_i) = (c_{1}^Y, c_{2}^Y, \cdots, c_{m_i}^Y, 0, \cdots, 0)V_i^T + b_k^c \quad (7)$$

We take this value into the $Y_k$, then we will get the $Y_k$.

$$Y_k = s_{k1}^Y \{c_{1}^Y(r_1)\psi(I_{i1}) + \cdots + c_{m_i}^Y(r_i)\psi(I_{im}) + b_k^c\}$$

$$+ s_{k2}^Y \{c_{2}^Y(r_2)\psi(I_{i2}) + \cdots + c_{n_k}^Y(r_i)\psi(I_{im}) + b_k^c\}$$

$$+ \cdots$$

$$+ s_{kn_k}^Y \{c_{n_k}^Y(r_{kn_k})\psi(I_{im}) + \cdots + c_{kn_k}^Y(r_{kn_k})\psi(I_{im}) + b_k^c\}$$

$$= (s_{k1}^Y c_{1}^Y(r_1), \cdots, s_{k1}^Y c_{m_i}^Y(r_i), 0, \cdots, 0)V_i^T + s_{k1}^Y b_k^c$$

$$+ (s_{k2}^Y c_{2}^Y(r_2), \cdots, s_{k2}^Y c_{n_k}^Y(r_{kn_k}), 0, \cdots, 0)V_i^T + s_{k2}^Y b_k^c$$

$$+ \cdots$$

$$+ (s_{kn_k}^Y c_{n_k}^Y(r_{kn_k}), \cdots, s_{kn_k}^Y c_{n_k}^Y(r_{kn_k}), 0, \cdots, 0)V_i^T + s_{kn_k}^Y b_k^c$$

Fig. 6. CRNN activation function and node computing
Algorithm 3. The value of \( Y \) is normalized by a unified order. E.g. one sorts the rules by the rule number. A new Boolean vector \( U \) from \( V \) is got according to the rule order. The final output value of class, \( Y_k \), is represented as follows.

\[
Y_k = \left( \begin{array}{c}
Y_1 \varepsilon V_1, s_2 Y_2, \ldots, s_n Y_n \\
\vdots \\
U_1 I \\
\vdots \\
U_n I 
\end{array} \right) = \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
b_1^k \\
b_2^k \\
\vdots \\
b_n^k 
\end{array} \right) + b_0^k (9)
\]

where \( I \) is a unit column vector, namely \( I^T = (1, 1, \ldots, 1) \).

According to the structure of the CRNN, \( U_i I \) is just the number of the items in the rule \( r_i \). 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 Equation (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.

\[
Y_k = \left( \begin{array}{c}
Y_1 \varepsilon V_1, s_2 Y_2, \ldots, s_n Y_n \\
\vdots \\
U_1 I \\
\vdots \\
U_n I 
\end{array} \right) = \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
b_1^k \\
b_2^k \\
\vdots \\
b_n^k 
\end{array} \right) + b_0^k (10)
\]

All the \( s \) and \( c \) are known for us and the \( U \) can be known if a 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 Equation 11.

\[
k_{predict} = \arg\max_k \{Y_k\}, k \in \{1, 2, \ldots, M\} (11)
\]

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 follows.

\[
Y_1 = \left( \begin{array}{c}
s_1 Y_1 c_1, s_2 Y_2 c_2, \ldots, s_n Y_n c_n \\
\vdots \\
U_1 I \\
\vdots \\
U_n I 
\end{array} \right) = \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
s_1 Y_1 c_1 \\
s_2 Y_2 c_2 \\
\vdots \\
s_n Y_n c_n 
\end{array} \right) = 0.435 (12)
\]

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

B. General Form of Rule-based Decision

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

\[
Y_k = \left( \begin{array}{c}
Y_1 c_1 Y_1, s_2 Y_2 c_2, \ldots, s_n Y_n c_n \\
\vdots \\
U_1 I \\
\vdots \\
U_n I 
\end{array} \right) = \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
U_1 I \\
U_2 I \\
\vdots \\
U_n I 
\end{array} \right) \left( \begin{array}{c}
s_1 Y_1 c_1 \\
s_2 Y_2 c_2 \\
\vdots \\
s_n Y_n c_n 
\end{array} \right)
\]

where \( U_i I \) is just the number of the items in the rule \( r_i \). 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 [5][6][15].

- We treat \( Y_k \) (sum of \( s_1 Y_1 c_1 \)) in another way rather than in sum expression. All \( s_1 Y_1 c_1 \) within \( Y_k \) are sorted in the descending order. The class label with first highest value \( s_1 Y_1 c_1 \) is selected which is the first method described in [5].
- 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 serves the similar purpose as the confidence when the data set size is big enough. The class label with the highest arithmetic average of Laplace accuracy of the top-\( K \) rules is regarded as the predicting tag in [15]. \( Y_k \) here is the weighted average of confidence (the support value in \( Y_k \) is the weight 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 \( s_1 Y_1 c_1 \) (\( \text{sup}(r) \cdot \text{conf}(r) \) for short) has the following equal relation: \( Y_k = \sum \text{sup}(r) \cdot \text{conf}(r) = \sum \text{sup}(p,c) \cdot \text{conf}(p) = \sum \text{sup}(p,c) \cdot \text{sup}(p) \). The above \( Y_k \) has similar function with the sum of \( \chi^2 \) for each rule in the group [6].

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 [16]. We have conducted the accuracy study on these data and compared CRNN with C4.5 [7] and CPAR [15]. 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 rule 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
Moreover, CRNN has two good characteristics. CRNN holds a general expression of the rule based decision methodology. The nodes in CRNN are all transparent for users meaning that a clear interpretation of hidden nodes in CRNN can be given while it is hard for traditional neural network.

CRNN introduces a new approach towards efficient and high quality classification model integrating the classification rules into neural network. Our further work will focus on the following two tasks. The bias of node computing model is set to zero in this paper, and we will do more study on how to learn the bias parameters by a linear computing method. From the evaluation results, we know that the rule set is critical for CRNN. Better rule set makes higher accuracy of CRNN. We will do more research work on mining effective rule set.

### REFERENCES


