A Fuzzy Classifier Based on Correlation Matrix Memories

EREN AYKIN, SIMON O’KEEFE
Computer Science Department
The University of York
York YO10 5DD
UNITED KINGDOM
erenaykin@gmail.com http://www.cs.york.ac.uk

Abstract: - This paper describes a binary neural network classifier that is able to make decisions based on fuzzy relational rule sets. Rule sets are extracted from a training data set and stored in a Correlation Matrix Memory (CMM). Such a classifier has many advantages including suitability for hardware implementations, fast matching, handling of missing or erroneous data and online learning. The main purpose of this paper is to demonstrate the suitability of the AURA library for building CMMs that perform fuzzy operations.

Key-Words: - Fuzzy Classification, Neural Networks, Correlation Matrix Memory, AURA

1 Introduction

Applications that use neural networks for fuzzy classification are well known. Most of these methods use neural networks separately, in order to adjust membership function shapes [1], [2]. There are also some methods that propose artificial neuron designs that are capable of processing fuzzy values [3], [4]. Designs based on classical neural network architectures that perform classification tasks have some disadvantages and limitations. Passing values among layers of processing units can be time consuming during classification, the systems may not be able to represent missing values, and it can be hard to add new relations after the training process is finished.

There have been many methods proposed for learning fuzzy relations from training data sets composed of crisp values [5], [6]. We use two methods for this purpose and compare their effects on the performance of our CMM architecture. The first method was proposed by Jain & Abraham [7]. It uses the mean and standard deviation values of samples of each class. Each new sample is assigned a membership degree to each class based on its similarity to common properties of corresponding classes. This approach allows creation of only one rule per class. The other method we use for generating fuzzy rules is to use NEFCLASS [8].

Our design uses CMM for storing and recalling fuzzy relations. CMM is a single layered neural network that uses binary weights for pattern association. We use the AURA C++ library [9] which provides functions and classes that can be used to implement CMM based applications.

Section 2 describes AURA and the CMM that lies at the heart of our design. Section 3 explains the fuzzy rule extraction methods and more precisely how they are used in our project. Section 4 focuses on the design we propose to be used for fuzzy classification. Section 5 shows results of some parameter tests and provides a comparison with the well known C4.5 decision tree generation algorithm.

2 The Advanced Uncertain Reasoning Architecture

AURA belongs to a group of neural networks that are called Random Access Memory (RAM) based neural networks. A detailed description of RAM based neural networks can be seen in [10]. They are also called weightless neural networks since they do not use real valued weights like most of the classical models. Weights are either 1 or 0, indicating the presence or non-presence of a connection.

AURA has been previously used for reasoning with crisp rules as a rule matcher, which matched crisp antecedent values with crisp consequent values [11]. This architecture required separate CMMs for rules with different arities.

2.1 The Correlation Matrix Memory

A CMM is a single layered network that implements pattern association. Input and output vectors are used to form a matrix, as explained below, and they are associated by setting the weights at corresponding intersection points to 1. This structure provides fast recall because there are no multiple layers of individual processing units. Furthermore time performance is independent of the previous information stored in the CMM, unlike
classical architectures where unit number and processing time increase as the amount of information to store increases.

2.1.1 Learning

The learning process is initialized by setting all weights to 0. Then, weights that correspond to intersection points where both input and output patterns have the value 1 are set to 1. This process is applied to each input-output pair.

\[
\text{Fig. 1: Learning in CMM. The input pattern is presented to the rows of the memory. If rows and columns of the matrix are thought as wires, weights that are placed at the intersection points where both rows and columns (columns correspond to the bits of the desired output) have the value 1 are set to 1 by connecting the corresponding wires.}
\]

The process is formulated as shown below. Outer products of the input and output binary vectors are bitwise or'ed, where, \( I \) and \( O \) represent the input and output binary vectors.

\[
CMM = \bigvee I \times O^T
\]

This approach completes training of a single pair of patterns in a single step

2.1.2 Recall

During the recall process mappings formed during training are invoked. When an input pattern is presented to the CMM, weights that correspond to set bits of the input pattern are added for each column as shown in Fig. 2.

\[
\text{Fig. 2: Recalling in CMM. The same patterns used for training in Fig. 1 are recalled. It is observed that highest output values correspond with set bits of output patterns.}
\]

The output value should be thresholded in order to get a binary vector. Two of the mostly used methods are Willshaw thresholding and L-Max thresholding. Willshaw thresholding [12] is performed by determining a fixed value of threshold, generally set to number of bits set in the input vector (typically the number of bits set are equal for all patterns in order to ensure a uniform density), and setting the values that are equal to the threshold value to ‘1’ while setting others to ‘0’. L-Max thresholding [13] is performed by determining a value for the threshold that sets the columns that have the highest \( L \) numbers to ‘1’. Willshaw method is preferred for cases when exact matching of input and output data is important, meanwhile L-Max method is preferred when erroneous or missing input data is possible and it is acceptable for a system to provide a sub-set of possible solutions.

This ability to do partial matching is an important advantage. If output thresholds are adjusted accordingly, results that are similar to the trained patterns can be retrieved.

2.2 Quantisation

CMMs operate on binary input data, therefore numerical values need to be quantised and encoded into binary values. Two of the methods proposed for unsupervised quantisation are equi-frequency binning and equi-width binning.

The main purpose of the equi-frequency method is to prevent saturation of the CMM, i.e. forming of a matrix with high densities of weights in some areas. The idea is that, if some of the bins receive more input, it is more likely that the rows that they are associated with them get saturated. To prevent this, boundaries of each bin are adjusted so that they receive about the same numbers of input values and weights are distributed uniformly. In other words, with this method each bin corresponds to an equal number of input patterns including the redundant ones [14].

The equi-width method is used in our system. It works as follows: An initial number of bins \( n_b \) is set for each attribute. For each attribute, redundant data is removed and the data is sorted in ascending order, therefore there are \( n_p = n / n_b \) input values for each bin, where \( n \) is the number of distinct attribute values. The value interval of each bin is
determined by left and right bounds of corresponding partitions when the sorted data is divided into \( n_p \) equal partitions. The quantisation process is shown in more detail in Fig.3.

Fig.3: Quantisation with equi-width and equi-frequency approaches. Bins of the equi-width approach are mapped with equal ranges of input values as shown with the red dashed line. Bins of the equi-frequency approach are mapped with variable ranges of input values in order to map each bin with an equal number of samples as shown with the blue dashed line.

It was shown that the equi-width method performs better for distance based applications (in their case, the k-NN clustering) because it restricts distortion of squared Euclidean distance approximation [15]. Since our application is also distance based, equi-width method is more suitable for it.

### 3 Fuzzy Rule Generation

A fuzzy rule for classification has the form:

\[
R: \text{If } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } ... \text{ and } x_n \text{ is } A_n \text{ then } y \text{ is } C_k
\]

where \( x_n \) represents a crisp attribute value, \( A_i \) represents a fuzzy set and the output \( y \) represents the membership degree of the sample to the class \( C_k \). This value indicates the possibility of the sample belonging to the relevant class. It is determined by degree of fulfillment of the antecedent part of a rule. Two methods for creating rules in this form from data sets with crisp values are investigated in this work. They are Jain & Abraham’s method and NEFCLASS.

#### 3.1 Jain & Abraham’s Method

In [7], Jain & Abraham propose a method for generating fuzzy rules with mean and standard deviation values of the samples that belong to the same class. The formula used to find membership degree of an attribute \( x_i \) to a fuzzy set \( A_i \) is given in Equation (3.1)

\[
A_i(x_i) = \exp \left(-\frac{(x_i - \mu_i)^2}{2(\sigma_i)^2}\right)
\]

Here, \( \mu_i \) is the mean value of the \( i \)th attribute of the patterns that belong to class \( k \) and \( \sigma_i \) is the standard deviation. Since this method produces only one rule for each class it may not perform well for cases when samples that belong to a certain class are grouped in many places in the data space.

#### 3.2 NEFCLASS

NEFCLASS (Neuro Fuzzy CLASSification) [8] is a neuro-fuzzy classifier for generating and tuning fuzzy rule sets. It uses backpropagation in order to adjust shapes of the fuzzy sets in rules. The architecture of the classifier is shown in Fig.4.

Fig.4: An example NEFCLASS Architecture. There are 3 input units for 3 attributes, 4 hidden units for a rule set of 4 rules and 2 output units for 2 classes. A fuzzy set membership is represented as \( j_i \), where \( \mu_i \) is the fuzzy set and \( j_i \) is the attribute number. For example the 2nd rule in the figure is: if \( I_1 \) is \( \mu_1 \) and \( I_2 \) is \( \mu_2 \) and \( I_3 \) is \( \mu_3 \) then \( y \) is \( c_1 \).

The architecture is similar to a multi layer perceptron, however weights are not real values but fuzzy membership functions.

With this method, the data space is partitioned into grids according to the previously determined number of fuzzy sets. Only the partitions that contain enough information are included in the rule
set. These partitions are then labeled with appropriate class names. Each rule defines required intervals of attribute values in order to be classified with a certain label. After rule learning comes parameter learning via backpropagation, where membership functions are adjusted by shifting or by enlarging or reducing their supports.

### 3.3 A Brief Comparison of the Methods

The first method provides faster training since rule set generation and classification can be performed in a single process. With the second method, a NEFCLASS classifier is created in a separate process in order to adjust shapes of membership functions. Another process is required in order to store the rule set into the CMM. On the other hand, the second method is more suitable for representing data sets that include objects that belong to the same class but not are clustered around only one point. The major reason for that is the first method can only create one rule for each class, based on mean values of each attribute.

### 4 Application

The main purpose of this work is to show that the AURA model can be used for reasoning with fuzzy values. Our classifier architecture is explained in more detail in this section.

#### 4.1 Design

The overall system takes pattern attributes as its input and produces rule membership values as its output. The number of columns is dependent on how many rules are used to represent the data set and the number of rows is dependent on how many attributes patterns have. Rule membership values then undergo further processing to determine a winning rule and its related class. If more than one rule indicating different classes have the highest membership value, the winner rule is selected with roulette wheel selection. A representation of the system is given in Fig.5.

![Fig. 5: Overall system architecture. Crisp attribute values are quantized into bins. Those bins that represent attribute values are matched with rule membership values. Finally a selection process is applied to active output bins to determine a winner rule.](image)

Quantisation of the attribute values is performed by equi-width binning. The same number of input bins is allocated for all attributes and each attribute value is mapped uniformly across the ranges of values that bins represent. If there are pattern values that are higher than the maximum or lower than the minimum values observed in the training data set, they are mapped with the highest bin or the lowest bin accordingly.

### 4.2 Learning

Rule sets and fuzzy set information learned by the rule generation methods are transferred into the CMM. This is done by getting responses given by a rule set to every possible input pattern in the data space as follows:

```plaintext
for each bin related with the 1st attribute
  for each bin related with the 2nd attribute
    ...
    for each bin related with the n th attribute
      Get rule membership values according to the rule set.
      Store the associations into the CMM where rule membership values are higher than \( thr_R \).
    end
  end
end
```

The rule threshold \( (thr_R) \) is an important parameter for decreasing the saturation of the CMM. It
5. Experimental Results

Classification success of the system based on the values of its parameters is examined with three data sets. The L-Max threshold is an important parameter that affects the number of active output bins that will enter the selection process. Another important parameter is the rule threshold which affects the number of associations to be stored in the CMM as shown in the previous section.

5.1 Parameter Tests for Jain & Abraham’s Method

Fig. 6 a and b show the effect of L-Max threshold on classification success rate. Success rates tend to decrease as the number of active output bins increase in the output layer. Fig. 6 c and d show the effect of Rule threshold on the success rate. In this case, success rates tend to decrease as saturation of the CMM decreases as a consequence of increasing the Rule threshold.

When NEFCLASS is used for rule creation, L-Max and Rule thresholds produce similar effects on the classification success rate.

![Graphs showing L-Max and Rule threshold effects](image)

5.2. Comparison of Performances

Classification success of our architecture is compared with the well known C4.5 classifier, which generates non-fuzzy classification rules. C4.5 is a decision tree generation algorithm developed by Ross Quinlan and it is well known and widely used for rule extraction and classification [16]. A decision tree also has the advantage of representing the knowledge in human-readable rule sets. Decision trees are formed of nodes that perform tests on attribute values and leaves that produce final decisions.

One of the key conclusions that can be made out of the results shown in Table 1 is that differences between training data set representation successes and test data classification successes are acceptable for most cases, therefore the matching between input and output data is done without significant information lost. Furthermore, the classification success rates, especially the rates for the breast cancer data set show potential for high reliability upon further optimization.
Table 1: Experimental results and comparisons with the C4.5 method. Results for each method show both the classification success when the same training data is used as test data (shown as Train) and the classification success when the separate test data set is used as test data (Shown as Test).

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Attr. #</th>
<th>Class #</th>
<th>Missing Values</th>
<th>Sample # (Test / Train)</th>
<th>JA Method Test</th>
<th>NEFCLASS Test</th>
<th>C4.5 Test</th>
<th>JA Method Train</th>
<th>NEFCLASS Train</th>
<th>C4.5 Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>4</td>
<td>3</td>
<td>No</td>
<td>50 / 100</td>
<td>78%</td>
<td>80%</td>
<td>86%</td>
<td>96%</td>
<td>90%</td>
<td>98%</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>2</td>
<td>No</td>
<td>200 / 568</td>
<td>64%</td>
<td>68.13%</td>
<td>67%</td>
<td>74.47%</td>
<td>75.0%</td>
<td>79%</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>9</td>
<td>2</td>
<td>Yes</td>
<td>149 / 550</td>
<td>91.28%</td>
<td>88.36%</td>
<td>92.61%</td>
<td>92.73%</td>
<td>93.3%</td>
<td>98.7%</td>
</tr>
</tbody>
</table>

6 Conclusion

We propose a fuzzy classifier model that uses the AURA model for storing fuzzy relations. We demonstrate that the AURA model can be used successfully for fuzzy inference applications. The architecture shows a promising potential if it is used with a suitable rule learning method. The matching mechanism is also suitable to be used for a fuzzy controller. The learning algorithm needs to be optimized in order to reduce the load of the learning process. Reducing the number of bins to be processed can lead to faster learning.

References: