A Granular Computing Approach to Improve Large Attributes Learning

Fengming M. Chang†
Department of Information Science and Applications
Asia University
Wufeng, Taichung 41354, Taiwan
paperss@gmail.com

Chien-Chung Chan
Department of Computer Science
University of Akron
Akron, OH 44325-4033, USA
chan@uakron.edu

Abstract – Based on the concept of granular computing, this article proposes a novel Boolean Conversion (BC) method to reduce data attribute number for the purpose of improving the efficiency of learning in artificial intelligence. Data with large amount of attributes usually cause a system freezes or shuts down. The proposed method combines large amount attributes to smaller ones by the way of Boolean method. Three data sets are used to compare the learning accuracies and efficiencies by Bayesian networks (BN), C4.5 decision tree, support vector machine (SVM), artificial neural network (ANN), fuzzy neural network (FNN, neuro-fuzzy), and Mega-fuzzification learning methods. Results indicate that the proposed BC method can improve the efficiency of machine learning and the accuracy is not worse.

I. INTRODUCTION

Artificial intelligence (AI) methods have been widely used in many fields recently for data prediction [1-14]. In many cases, data used for AI researches have small number of attributes. However, in the real world, data sets usually have large number of attributes. This condition causes large calculation work problems in performing AI computation and computers are easy to be hold or shut down. In different kinds of learning methods, artificial neural network (ANN), fuzzy neural network (FNN, neuro-fuzzy), and FNN based mega-fuzzification [9, 15-16] need large computation works and are easy to cause a computer freezes. In our experiment, the number of data attributes exceeds 6, and fail to run to the end for some data sets. However, from the view point of machine learning, 6 is not a large number. The purpose of the proposed BC method is to solve above problem in FNN and FNN based mega-fuzzification. Nevertheless, other machine learning methods are also used to compare the results of BC. Such methods include Bayesian networks (BN), C4.5 decision tree, support vector machine (SVM), and ANN learning methods.

The rest of the paper is organized as follows. Section II reviews the related works. The proposed BC method is described in Section III. The computational results by three cases are presented in Section IV. Finally, conclusions are given in Section V, and it is followed by references.

II. RELATED WORKS

Data attribute reduction is an important way to improve the efficiency of AI learning. Early related work was done by Shen and Chouchoulas, who proposed a Rough Set Attribute Reduction (RSAR) method to remove redundant input attributes for discrete values from complex systems. However, RSAR still lacks efficiency although it can reduct attributes [18]. The other study is that Beynon introduced an “approximate reducts” concept and proposed a Variable Precision Rough Sets (VPRS) model to find out the smallest set of attributes [19]. Later, Hsu et al. applied VPRS model for mobile phone test procedure [20]. Inbarani et al. also applied VPRS for feature selection of web usage mining [21]. In addition, Ang and Quek did not reduce data attribute but reduce fuzzy rules by combined rough set and neuro-fuzzy learning [22].

III. BOOLEAN CONVERSION

A PUSTAK data set that was taken from the Rose2 software sample database [23] is used to explain the proposed BC method first. PUSTAK has 10 input and 1 output attributes. It can not be performed well in FNN and mega-fuzzification methods. Values of its 11 attributes are integers. Values of the first input to the ninth input attributes are from 1 to 6, of the tenth input are from 1 to 5, and values of output are 1 and 2.

†This work is partially supported by NSC 96-2416-H-468-006-MY2.
Corresponding author.
The process of the BC method is simple. Each decimal number can be transferred into a Boolean number one on one mapping. On the top of the Fig. 1, five decimal numbers, 5, 1, 1, 2, and 2, are transferred into five Boolean numbers accordingly. Considering the maximum value of each attribute, Boolean number for decimal number 1 should be 001, 2 should be 010, and so on. So that the number of bit in Boolean number for each attribute is fixed. Next, the five Boolean numbers are physically combined to be a unique Boolean number as shown in the middle part of the Fig. 1. Each original Boolean number occupies its own digital position in the combined Boolean number format without mixing with other numbers. After that, for the convenience of calculation in the real world, this combined Boolean number is transferred to a decimal number. In the above process, the five input values are combined into a unique value and the input attributes are reduced. The reason for not combining the decimal number directly, such as combine 5, 1, 1, 2, 2 to be 51122 is because that 51122 is bigger than 21074, the result of BC. Smaller number is easier for calculation.

![Diagram of the process of the Boolean conversion](image)

Fig. 1. The process of the Boolean conversion.

With Fig. 1 as an example, there are 5 inputs were converted into a single new input. First, the original 5 inputs {5, 1, 1, 2, 2} are converted into a Boolean digit number that is {101, 001, 001, 010, 010}. Second, these 5 Boolean digit numbers are physically combined into one Boolean digit number: 101001001010010. The corresponding decimal number is 21074. It can be expressed by the binary system as:

\[
\begin{align*}
\{101, 001, 001, 010, 010\} &
\rightarrow 101001001010010 \\
&= 101 * 100000000000 \\
&+ 001 * 10000000000 \\
&+ 001 * 10000000 \\
&+ 010 * 1000 \\
&+ 010 * 1
\end{align*}
\]

It could be a Boolean weight expressed by binary system as:

\[
B = [10000000000 1000000000 1000000 10000 1]
\]

or expressed by decimal system:

\[
[5, 1, 1, 2, 2] \rightarrow 21074 = 5 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 2 \times 2^1 + 2 \times 2^0
\]

and the binary weight vector is

\[
B = [2^5, 2^4, 2^2, 2^1, 2^0]
\]

the power of 2 is determined by the data maximum domain. For example, if the value range of the second attribute is from 1 to 6, the maximum domain is 6, then

\[
6(\text{decimal}) = 110(\text{binary}) = 2^2 + 2^1 + 2^0(\text{decimal}) \leq 2^5(\text{decimal})
\]

2^3 means the number of bits the second attribute needs is 3, bits(2) = 3. As well as the third to the fifth attributes, each one needs 3 also. This is a special condition in this case that all the attributes need 3, but each attribute may not need the same number of bits in other cases. The power of the second attribute is the bit number plus the power of the third attribute or the total bit number from the third to the fifth inputs. It is presented as:

\[
power(i) = \text{bits}(i+1) + power(i+1) = \sum_{m=1}^{i} \text{bits}(m)
\]

\[
1 \leq i \leq (I-1), \quad power(I) = 0,
\]

where \( I \) is the total number of the input \( i \).

IV. Computational Results

A. Case 1: PUSTAK

PUSTAK data that is described in section III is used by the proposed method in this subsection. TABLE I shows one record of the data. There are 10 input and one output attributes in the data. The 1st to the 5th attributes are converted into a new input attribute, the 6th to the 10th attributes are combined into another new input by BC. Therefore there are only two new input attributes. As shown in TABLE I, the new input record is (21074, 8780).

After all attributes are converted using BC, the data are tested and compared using BN, C4.5, SVM, ANN, FNN, and Mega-fuzzification methods with 10-folds cross-validation testing. Each fold are used as testing data in turn and the remaining total of 9 folds data are used as training data. The results are presented in TABLE II. Without using BC, FNN and Mega-fuzzification fail to perform. After applying BC, it can easily perform machine learning using FNN and Mega-fuzzification methods. The prediction accuracies after using BC are a little lower than without using it in this case, but learning time decreases. Fig. 2 compares the prediction
accuracies under different learning methods, and Fig. 3 compares the learning time needed.

<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
<th>#7</th>
<th>#8</th>
<th>#9</th>
<th>#10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original decimal value</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Converse to Boolean value</td>
<td>101 001 001 010 010 010 001 001 001 100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combine to two Boolean value</td>
<td>101001001010010 10001001001100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Converse to two decimal value</td>
<td>21074 8780</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>Method</th>
<th>Bayesian</th>
<th>C4.5</th>
<th>SVM</th>
<th>ANN</th>
<th>FNN</th>
<th>Mega-fuzzification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-BC</td>
<td>Accuracy</td>
<td>86.11%</td>
<td>94.91%</td>
<td>90.74%</td>
<td>89.81%</td>
<td>Fail to perform</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0.05</td>
<td>0.03</td>
<td>0.41</td>
<td>1.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>Accuracy</td>
<td>83.80%</td>
<td>88.43%</td>
<td>85.41%</td>
<td>87.96%</td>
<td>79%</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0</td>
<td>0</td>
<td>0.16</td>
<td>0.3</td>
<td>4.2</td>
<td>4.2</td>
</tr>
</tbody>
</table>

**TABLE III**

<table>
<thead>
<tr>
<th>Method</th>
<th>Bayesian</th>
<th>C4.5</th>
<th>SVM</th>
<th>ANN</th>
<th>FNN</th>
<th>Mega-fuzzification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-BC</td>
<td>Accuracy</td>
<td>56.09%</td>
<td>50.68%</td>
<td>55.46%</td>
<td>52.70%</td>
<td>Fail to perform</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0.01</td>
<td>0.06</td>
<td>41.7</td>
<td>257.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>Accuracy</td>
<td>53.38%</td>
<td>54.73%</td>
<td>53.38%</td>
<td>55.41%</td>
<td>53%</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0.01</td>
<td>0.02</td>
<td>41.5</td>
<td>29.5</td>
<td>4.7</td>
<td>4.8</td>
</tr>
</tbody>
</table>

**Fig. 2. The accuracy comparison before and after using BC method by six methods for PUSTAK data.**

When classifying output to 1 healthy and 4 sick degrees, the learning accuracies and time are presented in TABLE III and FIG. 4. In this case, not all learning accuracies after using BC are lower than before. When classifying output to healthy and sick two values, the learning accuracies and time are presented in TABLE IV and FIG. 5. In this case, all learning accuracies after using BC are higher than before. Such results are different from the PUSTAK case.

Still, before using BC method, learning in FNN and mega-fuzzification methods fails in heart disease data.

**B. Case 2: Heart disease**

The heart disease data are from Cleveland Hospital provided by Dr. Detrano (see UCI Machine Learning Repository [24]). There are in total 303 patients’ records in the data. Each record of the data has 13 inputs and one output attribute. Some items’ values in the data are missing, we got 296 records of data.

The total 14 attributes include 8 symbolic and 6 numeric items. For the convenience to some learning methods, symbolic attributes are transferred to numeric before learning. The 14 attributes with their values are list as the following: Age, Sex (1,0), Chest pain type (angina, abnang, notang, asympt) or (1-4), Trestbps (resting blood pres), Cholesterol, Fasting blood sugar < 120 (true or false) or (1,0), Resting ecg (norm, abn, hyper) or (0,1,2), Max heart rate, Exercise induced angina (true or false) or (1,0), Oldpeak, Slope (up, flat, down) or (1,2,3), Number of vessels colored, Thal (norm, fixed, rever) or (3,6,7), Either healthy (buff) or with heart-disease (sick) or (0 is healthy, 1,2,3,4 is sick).

There are two kinds of output classifications used. One is to classify output into healthy and sick, two values, the other is to classify output into healthy and four levels of sick, total of five values. It seems that to classify output only healthy and sick is more reasonable and accurate than classify into five values in our experiments later.

**TABLE IV**

<table>
<thead>
<tr>
<th>Method</th>
<th>Bayesian</th>
<th>C4.5</th>
<th>SVM</th>
<th>ANN</th>
<th>FNN</th>
<th>Mega-fuzzification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-BC</td>
<td>Accuracy</td>
<td>56.09%</td>
<td>50.68%</td>
<td>55.46%</td>
<td>52.70%</td>
<td>Fail to perform</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0.01</td>
<td>0.06</td>
<td>41.7</td>
<td>257.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>Accuracy</td>
<td>53.38%</td>
<td>54.73%</td>
<td>53.38%</td>
<td>55.41%</td>
<td>53%</td>
</tr>
<tr>
<td>Time(sec)</td>
<td>0.01</td>
<td>0.02</td>
<td>41.5</td>
<td>29.5</td>
<td>4.7</td>
<td>4.8</td>
</tr>
</tbody>
</table>

**Fig. 3. The learning time comparison before and after using BC method by six methods for PUSTAK data.**

When classifying output to 1 healthy and 4 sick degrees, the learning accuracies and time are presented in TABLE III and FIG. 4. In this case, not all learning accuracies after using BC are lower than before. When classifying output to healthy and sick two values, the learning accuracies and time are presented in TABLE IV and FIG. 5. In this case, all learning accuracies after using BC are higher than before. Such results are different from the PUSTAK case.

Still, before using BC method, learning in FNN and mega-fuzzification methods fails in heart disease data.
TABLE IV
THE COMPARISON OF HEART DISEASE DATA BY CLASSIFYING OUTPUT TO HEALTHY AND SICK TWO VALUES.

<table>
<thead>
<tr>
<th>Method</th>
<th>Bayesian</th>
<th>C4.5</th>
<th>SVM</th>
<th>ANN</th>
<th>FNN</th>
<th>Mega-fuzzification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-BC Accuracy</td>
<td>62.09%</td>
<td>58.65%</td>
<td>57.03%</td>
<td>57.03%</td>
<td>Fail to perform</td>
<td></td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.02</td>
<td>0.07</td>
<td>0.76</td>
<td>201.3</td>
<td>Fail to perform</td>
<td></td>
</tr>
<tr>
<td>BC Accuracy</td>
<td>73.31%</td>
<td>71.96%</td>
<td>69.59%</td>
<td>67.91%</td>
<td>73.05%</td>
<td>74.13%</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.01</td>
<td>0.01</td>
<td>0.42</td>
<td>9.9</td>
<td>4.3</td>
<td>4.4</td>
</tr>
</tbody>
</table>

Fig. 6. The accuracy comparison before and after using BC method by six methods for Hayes-Roth data.

V. CONCLUSIONS

When number of data attributes is large, it becomes an obstacle in machine learning. Some leaning methods need a large of computation works, and large amount attributes causes a system easy to freezes. On the other hand, some leaning systems have their limits and can not allow data have too many attributes. Therefore, this study proposed a granular computing based Boolean Conversion method to reduce data attributes by a binary transfer process. After attributes are combined and reduced, with three data sets, we compare the learning results using BN, C4.5, SVM, ANN, neuro-fuzzy, and Mega-fuzzification learning methods. In some cases, learning accuracies after using BC are a little lower than before, some cases have higher accuracies, and some cases have both higher and lower accuracies. In general, the learning accuracy after applying BC is not worse. In addition, learning time is shortened after BC is used. Facing the problem of “fail to perform” in neuro-fuzzy, the proposed BC method indeed solves the problem of data have large attributes in learning in brief.

ACKNOWLEDGMENT

Thanks are due to the support in part by the National Science Council of Taiwan under Grant No. NSC 96-2416-H-468-006-MY2.

REFERENCES


