Data Attribute Reduction using Binary Conversion

FENGMING M. CHANG
Department of Information Science and Applications
Asia University
Wufeng, Taichung 41354
Taiwan
paperss@gmail.com

Abstract: - While learning with data having large number of attribute, a system is easy to freeze or shut down or run for a long time. Therefore, the proposed Binary Conversion (BC) is a novel method to solve this kind of large attribute problem in machine learning. The purpose of BC is to reduce data dimensions by a binary conversion process. All the attributes are reserved but combined into few numbers of new attributes instead of that some attributes are removed. To prevent the information loss problem during the conversion, each binary type data value occupies its own digital position in BC. In addition, 4 data sets: nbuses, ACLP, MONK3, and Buseskod data are used in this study to test and compare the learning accuracies and learning time. The results indicate that the proposed BC can keep about the same level of accuracy but increase the learning efficiency.

Key-Words: - Binary conversion, Large attribute, Machine learning, Neuro-fuzzy, Mega-fuzzificaiton

1 Introduction

Recent have shown widely applications in the fields of Artificial intelligence (AI) for data classification or prediction [1-14]. Many methods were proposed. For most of the examples in previous studies, the number of input attributes is not large. They probably only provide a theoretic model for researchers. However, real data in some theoretic studies and some practical applications have plenty of input attributes. It causes some problems. First, some systems will easily shut down because the calculations of the machine learning are too large. Second, some learning programs have their limits. The learning methods that mostly need to reduce input attribute numbers are Artificial Neural Network (ANN), Fuzzy Neural Network (FNN, neuro-fuzzy), and Mega-fuzzification, the later is improved based on FNN. FNN and Mega-fuzzification are more difficult to perform than ANN. FNN deals with the network learning using fuzzy membership functions. Because the defuzzification calculations in FNN are complex and difficult, most of the time fuzzy membership functions are setup as triangular, generalized bell, trapezoidal, and so on so that they are easy to calculate. Anomalous shapes fuzzy membership is not recommended because it is almost impossible to defuzzify using programs beforehand even though the defuzzy calculation is still not efficient. When the input attribute amounts are larger than 6, the FNN program could not perform normally. Most of the time, the computer went on hold without any response, it froze. In this article, nbuses data set has 9 input attributes. These data are used as "fail for learning" when using FNN or Mega-fuzzification methods because they have too many attributes. On the other hand, some machine learning programs have upper limit of network nodes, and reduction of the input attribute amounts is also necessary.

2 Literatures Review

The relative works and machine leaning methods that are used for comparison are reviewed in this section.

2.1 Data attribute reduction

It is a way to improve machine leaning efficiency by reducing number of data attributes. In the relative researches, Shen and Chouchoulas proposed the first data attribute reduction method, a Rough Set Attribute Reduction (RSAR) method, to remove redundant input attributes for discrete values from complex systems. Besides, an "approximate reducts" concept was offered by Beynon. In the mean time, he also proposed a Variable Precision Rough Sets (VPRS) model to find out the smallest set of attributes [16]. Afterwards, VPRS was

applied in other studies, such as Hsu et al. used VPRS model for mobile phone test procedure [17], and Inbarani et al. used VPRS for feature selection of web usage mining [18]. In addition, although Ang and Quek did not reduce data attriubute but reduce fuzzy rules by combined rough set and neuro-fuzzy learning to improve machine learning efficiency[19].

2.2 Bayesian networks [23]

Bayesian networks (BN) is a hierarchical classification method. They are consists of graph and probability theory as graphical models [22]. BN is defined as a directed acyclic graph or a probabilistic graphical model that presents a set of variables and their causal influences. The causal dependencies between the variables are expressed by conditional probability distributions. It usually set up numeric values with normal or Gaussian distribution in BN [23].

2.3 Decision tree ID3 and C4.5 [23]

ID3 is another hierarchical classification method to be a decision tree [24]. A decision tree performs categorical split following the value number of input attributes. A decision tree is a predictive model mapping from observations to predict the target values. Written by C, C4.5 is an improved version of ID3. Both symbolic and numeric values of input attributes can be classified and then outputs a classification tree.

2.4 Support vector machine [23]

Basically SVM separates data points into two groups of linear separable data sets. It tries to find out the optimal hyperplane by minimizing an upper bound of the errors and maximizing the distance, margin, between the separated hyperplane and data [25]. It was first invented for binary classification problems based on statistical theory. A maximal separating hyperplane is built by SVM to map input vectors to a higher dimensional space. Two parallel hyperplanes are built and the data are separated on each side of the hyperplane. Given training dataset (x_i, y_i) , $i = 1, \ldots, k$, where $x_i \in R^n$ and $y \in \{1, -1\}^k$, SVM tries to find out the optimal solution problem using the following form mainly:

$$\min_{w,b,\varepsilon} \ \frac{1}{2} w^T w + C \sum_{i=1}^{k} \varepsilon_i$$

subject to

$$y_i (w^T \varphi(x_i) + b) \geq 1 - \varepsilon_i, \ \varepsilon_i \geq 0$$

2.5 Artificial neural network [23]

An ANN consists of nodes that are interconnected using mathematical model with computational model. It computes inputs data by above models to gain its output. In more practical terms, an ANN is a non-linear tool. It can model complex relationships between inputs and output data. In most of times, an ANN is an adaptive system that can adjust the parameters to improve its performance for a given task. There are three types of neural network learning algorithms: supervised, reinforcement, and unsupervised leaning. Back-propagation neural network (BPN) is the best known supervised learning algorithm.

2.6 Neuro-fuzzy [26]

The ANFIS [10], a fuzzy neural network (FNN) tool, performs neural learning using fuzzy typed numbers. Given a set of input and output data, the ANFIS can constructs a fuzzy inference system with membership functions values adapted using a backpropagation algorithm or in combination with a least square method. The adaptation function of the ANFIS provides the machine learning system with FNN characters.

The basic model of the ANFIS is the Sugeno fuzzy model [12]-[14]. In the model, assuming x and y are two input fuzzy sets and z is the output fuzzy set, and the fuzzy if-then rules is formatted as:

If $x = P$ and $y = Q$ then $z = f(x, y)$

Consider two first-order rules of Sugeno fuzzy model, the if-then rules can be:

Rule A: If $x = P_a$ and $y = Q_a$, then

$$f_a = m_a x + n_a y + c_a,$$

Rule B: If $x = P_b$ and $y = Q_b$, then

$$f_b = m_b x + n_b y + c_b$$

where P_a, Q_a, P_b, and Q_b are fuzzy set values and m_a, n_a, c_a, m_b, n_b, and c_b are constants. In figured presentation, Fig. 3 shows ANFIS structure with a five-layer artificial neural network.

Denote that the output of the ith node of layer l are $O_{l,i}$. In Layer 1 of the ANFIS,

$$O_{1,i} = \mu_{M_i}(x), \ i = 1, 2, \text{ or}$$

$$O_{1,i} = \mu_{N_{i-2}}(y),\ i=3,4$$

where μ_{M_i} and $\mu_{N_{i-2}}$ are fuzzy membership functions that can be any membership type such as triangular or generalized bell function.

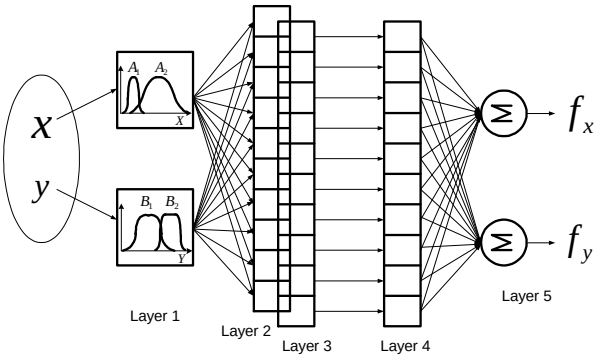

Fig. 3. The ANFIS structure.

For nodes in Layer 2, the outputs w_i are the product of the outputs of layer 1 and are used as the weights of Layer 3:

$$O_{2,i} = w_i = \mu_{M_i}(x)\mu_{N_i}(y),\ i=1,2$$

In Layer 3, the output of every node is normalized by a calculation as the following:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2},\ i=1,\ 2.$$

Next, Layer 4 is the defuzzy layer which adapts node values with equation:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (m_i x + n_i y + c_i),\ \text{for } i=1,\ 2$$

where m_i, n_i, and c_i are consequent parameters of the nodes.

Finally, the fifth layer is to compute the output of all the input signals using the equation:

$$O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i},\ \text{for } i=1,\ 2$$

2.7 Mega-fuzzificztion [9, 27, 28, 29, 30, 31]

Mega-fuzzification method was proposed for the purpose of solving the prediction of the best strategy problem in the Flexible Manufacturing System (FMS) when data are small [2, 4, 6]. In the studies of [2, 4, 6] for the Mega-fuzzification, several concepts were offered, such as data fuzzified, continuous data band, domain external expansion, and data bias adaptation.

The concept of the continuous data band was first proposed by Li, et al. [2]. Such a data continuing technology aims to fill the gaps between individual data and make incomplete data into the more complete status as presented by Huang and Moraga [11-12].

Furthermore, in the studies of Li, et al. and Chang [2, 4, 6], the domain range external expansion concept was also proposed into the procedure of the continuous data band method. In addition to filling the data gaps within the data set, the purpose of domain external expansion is also to add more data outside the current data limits, because possible data are expected not only inside but also outside the current data range.

To fill the gaps between crisp data, crisp learning data are transformed into continuous data as Figure 1 illustrates. In Fig. 2, there are five original crisp data. When these data are transformed into continuous type, virtual data between the crisp data are thus generated.

Fig. 2. Crisp data are transformed into continuous

The data effects have to be estimated also. In this research, continuous data are presented in fuzzy membership function forms. The fuzzy membership function can be a general bell, triangle, or even anomalous type and is the data's effective weight in the FNN learning later. Most of the time, an asymmetric fuzzy membership function is initiated. For example, a triangular fuzzy membership can be:

$$\mu_{\tilde{A}}(x_i) = \begin{cases} 0 & , x_i < \min \\ \dfrac{x_i - \min}{mid - \min} & , \min \le x_i \le mid \\ \dfrac{\max - x_i}{\max - mid} & , mid \le x_i \le \max \\ 0 & , x_i > \max \end{cases}$$

where min is the minimum value and max is the maximum value of the crisp learning data, and $mid = \frac{\min + \max}{2}$.

When crisp are transformed into continuous, boundaries of the continuous data band are minimum and maximum value of the original crisp data. However, the real data range is possible outside this data band. In order to build up real knowledge, the data band is externally expanded to the possible range in this study as illustrated in Fig. 3.

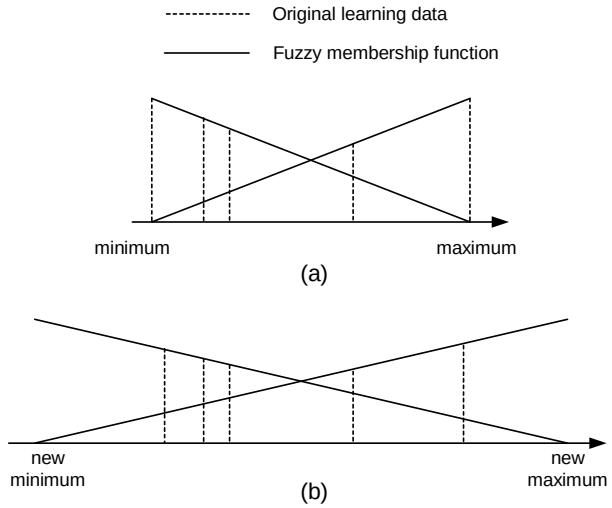

Fig. 3. Two triangular membership function values: (a) before, (b) after domain range external expansion

Because data have been transformed into a continuous data band, FNN are used in this study. During the learning process, the shape of fuzzy membership function is adapted to fit the best learning results.

3 The Proposed Method

First, a data set of nbuses [20] that consists of 9 input and 1 output attributes is used to explain the proposed BC method first. The nbuses can not be performed well in FNN and mega-fuzzification methods. Values of its 10 attributes are integers. The values of the first attribute are {1, 2, 3, 4}, values of the second, the 8th, and output attributes are {1, 2, 3}, and values of the other attributes are {1, 2}.

The process of the BC method is simple. Each decimal number can be transferred into a Boolean number one on one mapping. For example, in the first instance of our nbuses data, 9 attributes are combined into 3 new attributes. We combine the first to the third attributes to be the first new attribute. On the left of the Fig. 4, 3 decimal numbers, 4, 3, and 1, are transferred into 3 Boolean numbers accordingly. Considering the maximum value of each attribute, Boolean number for decimal number 1 should be 01. So that the number of bit in Boolean number for each attribute is fixed. Next, the 3 Boolean numbers are physically combined to be a unique Boolean number as shown in the middle part of Fig. 4. 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 3 input values are combined into a unique decimal value 77 and the input attributes are reduced. The reason for not combining the decimal number directly, such as combine 4, 3, 1 to be 431 is because that 431 is bigger than 77, the result of BC. Smaller number is easier for calculation.

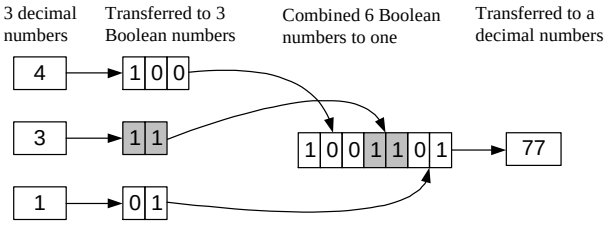

Fig. 4. The process of the Boolean Conversion.

With Fig. 4 as an example, there are 3 inputs were converted into a single new input. First, the original 3 inputs {4, 3, 1} are converted into a Boolean digit number that is {100, 11, 01}. Second, these 3 Boolean digit numbers are physically combined into one Boolean digit number: 1001101. The corresponding decimal number is 77. It can be expressed by the binary system as:

$$\begin{aligned} &\{\mathbf{100, 11, 01}\} \\ &\rightarrow 1001101 \\ &= \mathbf{100} * 10000 \\ &+ \mathbf{11} * 100 \\ &+ \mathbf{01} * 1 \end{aligned}$$

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

$$\boldsymbol{B} = [10000 \;\; 100 \;\; 1]$$

Table 1. An explanation of converting 9 attributes to 3 new decimal attributes.

	#1	#2	#3	#4	#5	#6	#7	#8	#9
Original decimal value	4	3	1	2	1	2	1	3	1
Converse to Boolean value	100	11	01	10	01	10	01	11	01
Combine to three Boolean value	1001101			100110			011101		
Converse to three decimal value	77			38			29		

or expressed by decimal system:

$$\{\mathbf{4},\mathbf{3},\mathbf{1}\} \rightarrow 77 = \mathbf{4} * 2^4 + \mathbf{3} * 2^2 + \mathbf{1} * 2^0$$

and the binary weight vector is

$$\boldsymbol{B} = [2^4 \quad 2^2 \quad 2^0]$$

4 Results and Comparisons

In this study, 4 data sets are offered to check learning accuracies and time of the results of both non-applying and applying BC. These data sets are nbuses, ACLP, MONK3 and Buseskod data sets.

4.1 nbuses data

The nbuese data that has been mentioned in section 3 is used by the proposed method in this subsection. Table 1 shows one record of the data. There are 9 input and one output attributes in the data. The 1st to the 3rd attributes are converted into a new input attribute, the 4th to the 6th attributes are combined into the 2nd new input by BC, and the 7th to the 9th are combined into the 3rd new one. Therefore there are only three new input attributes. As shown in Table 1, the new input record is {77, 38, 29}.

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 2. 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. Most of the prediction accuracies after using BC are even a little higher than without using it in this case, and learning time decreases. Fig. 5 compares the prediction accuracies under different learning methods. Fig. 6. illustrates the learning time. For nbuses data, even after using BC, time for FNN and mega-fuzzification is still large. For other learning methods, learning time reduced.

4.2 ACLP data

There are in total 140 instances in ACLP data with 6 input and 1 output attributes. The 1st, the 5th, and the 6th attributes have values of {1, 2, 3}, and the other attributes have values of {1, 2}. For neuro-fuzzy learning, 6 input attributes cause learning process very slowly. Fortunately, the values of each attribute are not large. We can compare the results of FNN and mega-fuzzification methods. The results are presented in Table 3, Fig. 7, and Fig. 8. Accuracies after using BC are a little lower than before, but time is saving. Before applying BC in FNN and mega-fuzzification, time for learning is very large. After using BC, time is saved largely.

Table 2. The comparison of nbuses data.

	Method	**Bayesian**	**C4.5**	**SVM**	**ANN**	**FNN**	**Mega-fuzzification**
Non-BC	Accuracy	93.42 %	93.42%	86.84%	85.53%	Fail to perform	Fail to perform
	Time(sec)	0.05	0.16	0.56	0.56		
BC	Accuracy	94.74%	93.42%	84.21%	92.11%	95%	95%
	Time(sec)	0	0	0.42	0.22	3	3

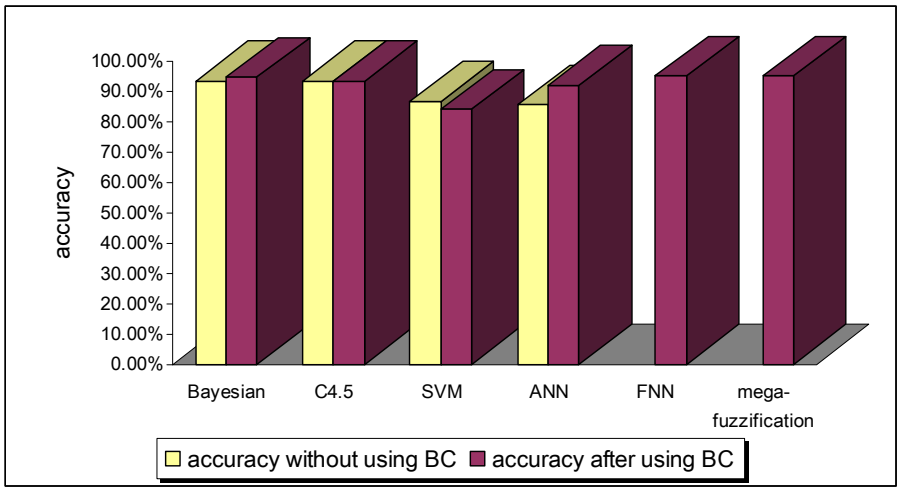

Fig. 5. The accuracy comparison before and after using BC method by six methods for nbuses data.

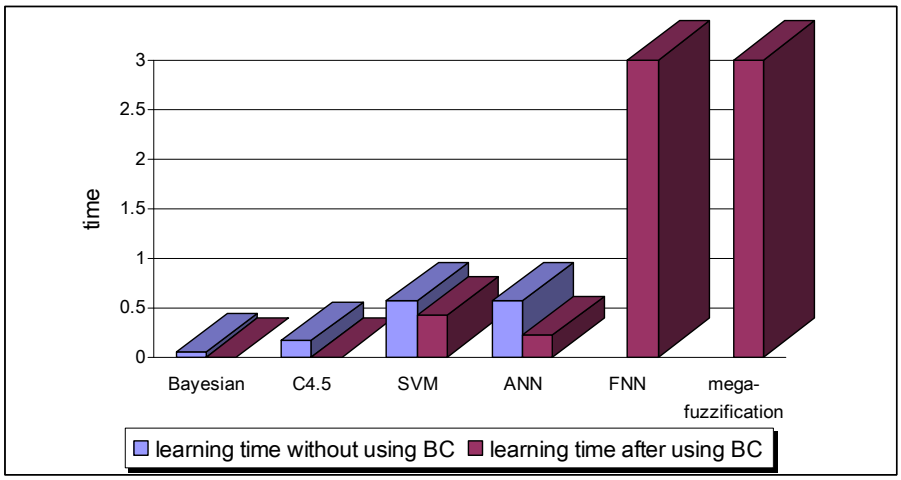

Fig. 6. The learning time comparison before and after using BC method by six methods for nbuses data.

Table 3. The comparison of ACLP data.

	Method	Bayesian	C4.5	SVM	ANN	FNN	Mega-fuzzification
Non-BC	Accuracy	86.43 %	87.86%	90.71%	84.29%	84.73%	84.73%
	Time(sec)	0	0	0.17	0.41	650	650
BC	Accuracy	80.71%	89.29%	81.43%	81.43%	80.38%	81.23%
	Time(sec)	0	0	0.02	0.2	5	5

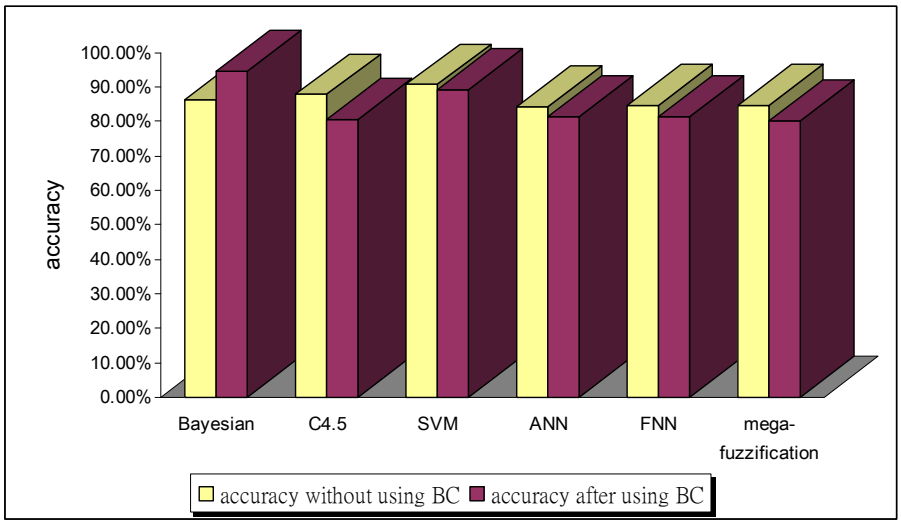

Fig. 7. The accuracy comparison before and after using BC method by six methods for ACLP data.

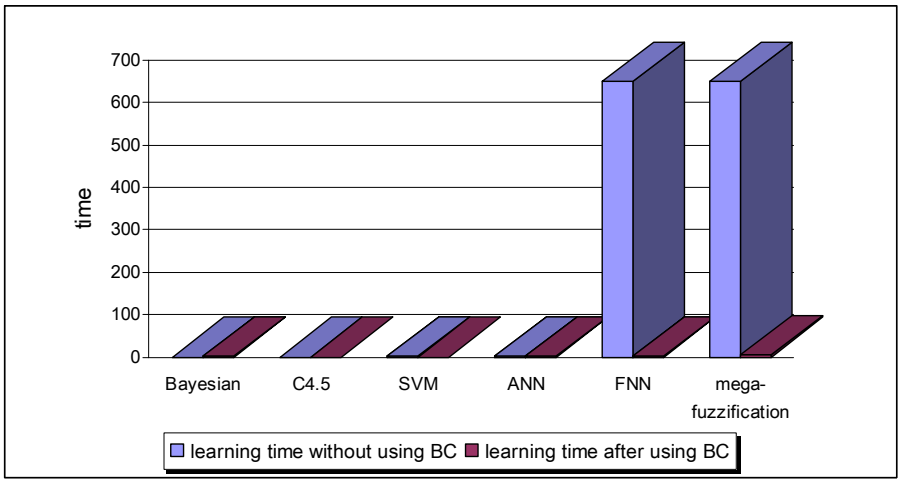

Fig. 8. The learning time comparison before and after using BC method by six methods for ACLP data.

4.3 Monk3 data

Monk3 data were created by Sebastian Thrun (see UCI Machine Learning Repository [21]) which has 432 instances, 6 inputs and 1 output attributes. Because the number of attributes is not large in this case, we can compare the learning accuracies of FNN and mega-fuzzificatioin with and without using BC again. Table 4 shows the results. In this case, FNN and mega-fuzzification can be performed but waste large time before using BC. All the accuracies after using BC are a little lower than before. The learning accuracies are also compared in Fig. 9 and learning time is compared in Fig. 10. Still, leaning time before using BC for FNN and mega-fuzzification is very large, but becomes very small after applying BC.

Table 4. The comparison of Monk3 data.

	Method	**Bayesian**	**C4.5**	**SVM**	**ANN**	**FNN**	**Mega-fuzzification**
Non-BC	Accuracy	92.36 %	100%	80.56%	100%	100%	100%
	Time(sec)	0	0.02	0.03	1.19	700	700
BC	Accuracy	89.21%	96.06%	76.90%	98.87%	97%	98%
	Time(sec)	0	0	0.02	0.61	10	10

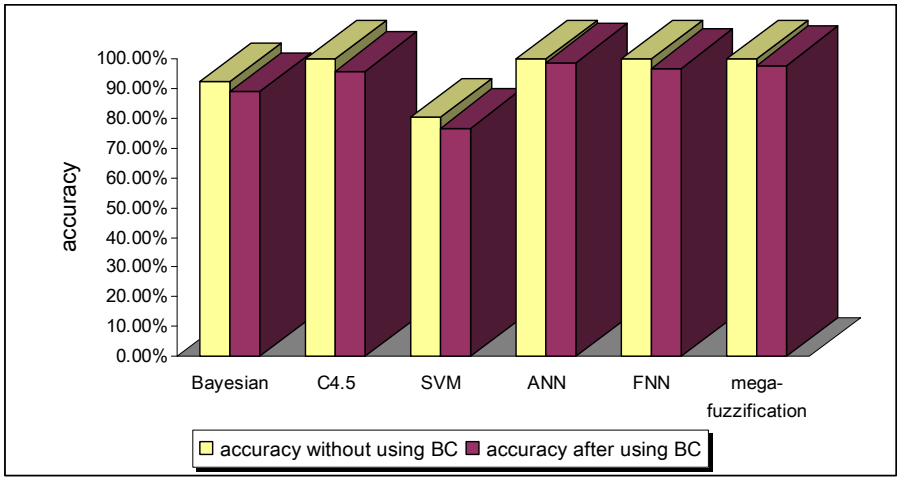

Fig. 9. The accuracy comparison before and after using BC method by six methods for Monk3 data.

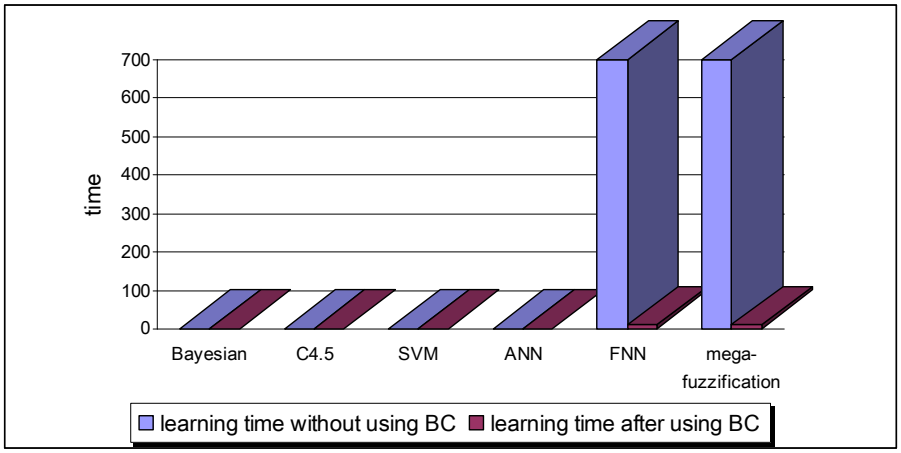

Fig. 10. The learning time comparison before and after using BC method by six methods for Monk3 data.

4.4 Buseskod data

There are 76 instances in Buseskod data set with 8 inputs and one output attributes. The values of the 1st to the 7th inputs are {0, 1} and the value ranges of the 8th input are {0, 1, 2} with the values of output {1, 2}.

Still, the learnings of FNN and Mega-fuzzification fail to perform before applied BC. The comparison results are shown in Table 5, Fig. 11, and Fig. 12. The learning accuracies after BC are a little worse or equal than those before BC, but the learning efficiencies are improved after BC.

Table 5. The comparison of Buseskod data.

	Method	Bayesian	C4.5	SVM	ANN	FNN	Mega-fuzzification
Non-BC	Accuracy	100%	98.68%	100%	100%	Fail to perform	Fail to perform
	Time(sec)	0.05	0.02	0.27	0.33		
BC	Accuracy	99.05%	98.68%	98.42%	100%	100%	100%
	Time(sec)	0	0	0.14	0.13	5	5

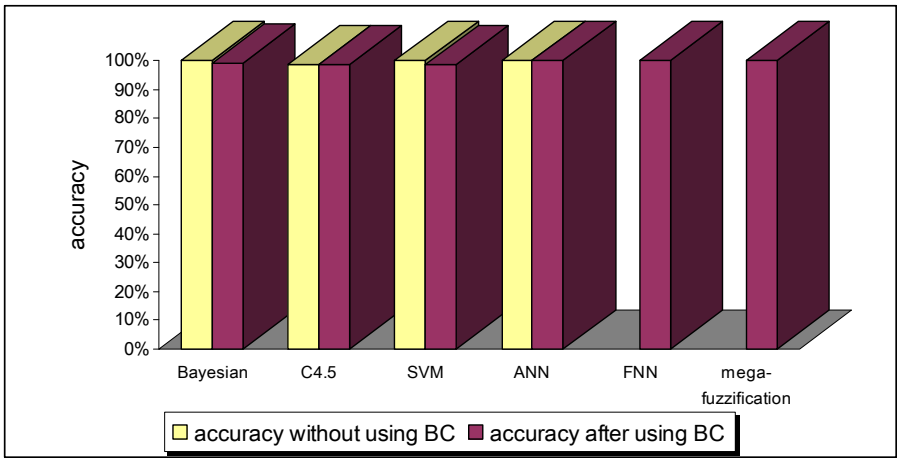

Fig. 10. The accuracy comparison before and after using BC method by six methods for Buseskod data.

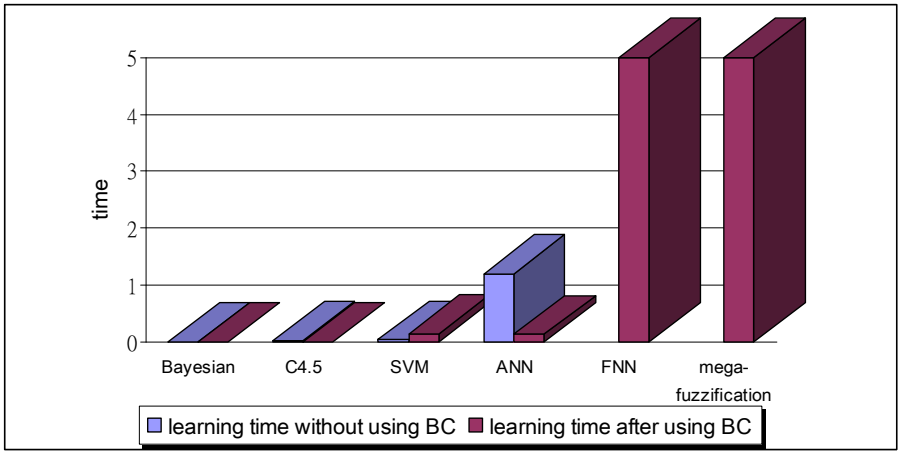

Fig. 12. The learning time comparison before and after using BC method by six methods for Buseskod data.

5 Conclusions

In this study, a novel BC method is proposed to deal with the problem of that data with a large number of attributes may cause a system freezes or shuts down. BC reduces attributes number by combining some of the attributes into smaller number of new attributes instead of that removing some attributes from data.

After attributes are combined and reduced, learning accuracies and learning time are compared by BN, C4.5, SVM, ANN, neuro-fuzzy, and Mega-fuzzification learning methods. Different learning methods have different learning characteristics, and a good learning method for one data is possible not suitable for another data [32]. The purpose of the comparison for neuro-fuzzy, such as BN, C4.5, SVM, and ANN, is to compare the learning accuracies and efficiencies by BC because FNN and Mega-fuzzification methods can not be performed in large attribute data.

The purpose of the proposed BC is not to improve learning accuracy but to solve the problem of failing to perform in large attribute data. Hence, learning accuracies are not improved in some data sets, but BC becomes a method for large attribute condition and learning accuracies are not too worse than before BC.

In this study, 4 data sets, nbuses, ACLP, MONK3, and Buseskod are used to test and compare the learning results. Some of their learning accuracies after using BC are a little lower than before, some have a little higher accuracies. In general, the learning accuracy after applying BC is not worse. In addition, leaning 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.

Acknowledgement

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:

[1] Y.Y. Yao, "Granular computing: basic issues and possible solutions," Proceedings of the 5th Joint Conference on Information Sciences, 1999, pp. 186 – 189.

[2] L. Polkowski and A. Skowron, "Towards adaptive calculus of granules," *Proceedings of 1998 IEEE International Conference on Fuzzy Systems*, pp. 111 – 116.

[3] T.Y. Lin, "Granular computing on binary relations I: data mining and neighborhood systems, II: Rough set representations and belief functions," in L. Polkowski and A. Skowron eds., *Rough sets in knowledge discovery 1*. Heidelberg, Physica-Verlag, 1998, pp. 107 – 140.

[4] Y.Y. Yao, "Granular computing using neighborhood systems," in R. Roy, T. Furuhashi , and P.K. Chawdhry (eds.) *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer-Verlag, London, 1999, pp. 539 – 553.

[5] T.Y. Lin, "Data mining: granular computing approach," *Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*, 1999, pp. 24 – 33.

[6] A. Skowron and J. Stepaniuk, "Information granules: towards foundations of granular computing," *International Journal of Intelligent Systems*, Vol. 16, 57 – 85, 2001.

[7] Y.Y. Yao, "Information granulation and rough set approximation," *International Journal of Intelligent Systems*, Vol. 16, 87 – 104, 2001.

[8] J.-S. R. Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference Systems," *IEEE Transactions on System, Man, and Cybernetics*, vol. 23, no.3, pp. 665-685, 1993.

[9] D. C. Li, C. Wu, and F. M. Chang, "Using data-fuzzifying technology in small data set learning to improve FMS scheduling accuracy," *International Journal of Advanced Manufacturing Technology*, Vol. 27, No. 3-4, pp. 321-328, 2005.

[10] F. M. Chang, and C. C. Chan, "Improve Neuro-Fuzzy Learning by Attribute Reduction," *The 27th Annual Meeting of the North American Fuzzy Information Processing Society*, The Rockefeller University, NY, USA, May 18-21, 2008.

[11] B. Predki, R. Slowinski, J. Stefanowski, R. Susmaga, and Sz. Wilk, "ROSE - Software Implementation of the Rough Set Theory," In: L. Polkowski, A. Skowron, eds, "Rough Sets and Current Trends in Computing," *Lecture Notes in Artificial Intelligence*, vol. 1424, pp. 605-608,. 1998.

[12] B. Predki and Sz.Wilk, "Rough Set Based Data Exploration Using ROSE System," In: Z. W. Ras, A. Skowron, eds, "Foundations of Intelligent Systems," *Lecture Notes in Artificial Intelligence*, Vol. 1609, pp.172-180, 1999.

[13] A. Øhrn and J. Komorowski, "ROSETTA: a rough set toolkit for analysis of data," *Proc. Third International Joint Conference on Information Sciences*, Vol. 3, pp. 403 - 407, Durham, NC, March 1997.

[14] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer, 1991.

[15] S. Qiang, and C. Alexios, "A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems," *Engineering Applications of Artificial Intelligence*, Vol. 13, No. 3, pp.263-278, 2000.

[16] M. Beynon, "Reducts within the variable precision rough set model: A further investigation," *European Journal of Operational Research*, Vol. 134, pp.592-605, 2001.

[17] J. H. Hsu, T. L. Chiang, and H. C. Wang, "VPRS model for mobile phone test procedure," *Journal of the Chinese Institute of Industrial Engineers*, Vol. 23, no. 4, pp.345-355, 2006.

[18] H. H. Inbarani, K. Thangavel, and A. Pethalakshmi, "Rough set based Feature Selection for Web Usage Mining," *International Conference on Computational Intelligence and Multimedia Applications*, pp.33-38, 2007.

[19] K. K. Ang, and C. Quek, "Stock Trading Using RSPOP: A Novel Rough Set-Based Neuro-Fuzzy Approach," *IEEE Transactions on Neural Network*, Vol. 17, no. 5, pp.1301-1315, 2006

[20] Laboratory of Intelligent Decision Support Systems, Poznan University of Technology, http://www-idss.cs.put.poznan.pl/site/rose.html

[21] UCI Machine Learning Repository, http://mlearn.ics.uci.edu/MLRepository.html

[22] E. J. M. Lauría, J. Duchessi, "A methodology for developing Bayesian networks: An application to information technology (IT) implementation," *European Journal of*

Operational Research Vol. 179, No.1, pp.234-252, 2007.

[23] F. M. Chang, "The Characteristics of Learning in Limited Data and the Comparative Assessment of Learning Methods," *WSEAS Transactions on Information Science and Applications*, Vol. 5, No.5, pp.407-414, 2008.

[24] J. R. Quinlan, "Learning decision tree classifiers," *ACM Computing Surveys* Vol. 28, No. 1, pp.71-72, 1986.

[25] K. Seo, "An application of one-class support vector machines in content-based image retrieval," *Expert Systems With Applications* Vol. 33, No. 2, pp.491-498, 2007.

[26] F. M. Chang, "Determination of the Economic Prediction in Small Data Set Learning," *WSEAS Transactions on Computers*, Vol. 5, No.11, pp.2743-2750, 2006.

[27] F. M. Chang and M. Y. Chiu, "A Method of Small Data Set Learning for Early Knowledge Acquisition," *WSEAS Transactions on Information Science and Applications*, Vol. 2, No.2, pp.89-94, 2005.

[28] F. M. Chang, "An intelligent method for knowledge derived from limited data," *Proceedings - 2005 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, pp.566-571.

[29] D. C. Li, C. Wu, T. I. Tsai and F. M. Chang, "Using mega-fuzzification and data trend estimation in small data set learning for early FMS scheduling knowledge," *Computers & Operations Research*, Vol. 33, No.6, pp. 1857-1869, 2006.

[30] D. C. Li, C. Wu, and F. M. Chang, "Using data continualization and expansion to improve small data set learning accuracy for early flexible manufacturing system (FMS) scheduling," *International Journal of Production Research*, Vol. 44, No.21, pp.4491-4509, 2006.

[31] F. M. Chang and Y. C. Chen, "A Frequency Assessment Expert System of Piezoelectric Transducers in Paucity of data," *Expert Systems with Applications*, Vol. 34, No.4, pp.2747-2753, 2008.

[32] M. Y. Kiang, "A comparative assessment of classification methods," *Decision Support Systems*, Vol. 35, pp.441-454, 2003.