Data Mining Using Dynamically Constructed Recurrent Fuzzy Neural Networks

Yakov Frayman and Lipo Wang

Deakin University, School of Computing and Mathematics,
662 Blackburn Road, Clayton, Victoria 3168, Australia
E-mail: yfraym@deakin.edu.au, lwang@deakin.edu.au

Abstract. Approaches to data mining proposed so far are mainly symbolic decision trees and numerical feedforward neural networks methods. While decision trees give, in many cases, lower accuracy compared to feedforward neural networks, the latter show black-box behaviour, long training times, and difficulty to incorporate available knowledge. We propose to use an incrementally-generated recurrent fuzzy neural network which has the following advantages over feedforward neural network approach: ability to incorporate existing domain knowledge as well as to establish relationships from scratch, and shorter training time. The recurrent structure of the proposed method is able to account for temporal data changes in contrast to both both feedforward neural network and decision tree approaches. It can be viewed as a gray box which incorporates best features of both symbolic and numerical methods. The effectiveness of the proposed approach is demonstrated by experimental results on a set of standard data mining problems.

1 Introduction

Data mining is one of the most promising fields for application of intelligent information processing technologies. A main aim of data mining is to extract useful patterns and nontrivial relationships from large collections of data records [3], [5], [10], [17]. This method can provide users with a powerful tool for exploiting vast amount of stored data.

The most popular approaches to data mining include symbolic [2], [7], [11] and neural [6], [9], [15] models. Symbolic models are represented as either sets of IF - THEN rules, or decision trees generated through symbolic inductive algorithms [2], [7], [11]. A neural model [6], [9], [15] is represented as an architecture of weighted nodes, each of which incorporates a thresholding function. While, in many cases, neural networks lead to more accurate classification results at the expense of long training time, it is difficult to incorporate available domain knowledge into neural networks [13], [14]. Traditionally, neural networks are not able to articulate knowledge in the form of classification rules [13], [14]; however, there exist techniques for extracting certain types of rules from trained neural networks [6], [9], [15].

The following characteristics of real world data may case difficulties to neural and/or symbolic learning techniques in data mining:

1. Temporal changes of data: During the database lifetime the contents of database can change. None of the existing symbolic or numerical approaches is able to account for this temporal changes of data, i.e., these classifiers need to be completely retrained as data change, which is undesirable in most real-world situations.
2. Need for data preprocessing: Most databases are not designed for data mining. Some existing approaches [1], [9] need special coding of the data before data mining. This may not be suitable for real-world situations as the user is usually not an expert on the method used for data preprocessing.
3. Use of existing knowledge: Whenever there exist expert knowledge on the database it is advantageous to be able to use it. Feedforward neural network approaches [9] are not able to incorporate the available expert knowledge.
4. Data volume: Million of records exist in many databases. Since the feedforward neural network [9] starts with a full-sized network, which is then pruned to an optimal size, the initial network can be too large in a large dimensional situation.

Fuzzy neural network (FNN) technique can be used as a bridge between numerical and symbolic data representations. Since fuzzy logic has an affinity with the human knowledge representation, it should become a key component of data mining system. One advantage of using fuzzy logic is that we can express knowledge about a database in a manner which is natural for people to comprehend.

In this paper we present a dynamically constructed FNN which can offer solutions to the abovementioned problems. Our FNN can reduce the computational cost as it starts with a minimal rule base which increases only when new input data requires so, at the same time constantly removing irrelevant inputs, rules, and rules conditions which no longer match the data. Dynamical construction of the network also eliminates the trial-and-error determination of the size of the hidden layer in feedforward neural networks [9] and creates a minimal network, thereby reducing the risk of overfitting. The proposed FNN does not need any preprocessing or postprocessing of the data as required in [1] and [9], which is desirable from the user's point of view. The recurrent structure of the proposed network is able to deal with temporal data changes without the requirement to retrain the classifier in contrast to both decision trees [11] and feedforward neural networks [9]. The presented FNN is able to incorporate *existing* domain knowledge in the form of IF-THEN rules, in contrast to feedforward neural networks [9].

X. Z. Wang *et al* [16] used fuzzy neural network for decision making. Our approach is different from X. Z. Wang *et al* [16] in the following aspects. Firstly, our FNN approach is recurrent on-line method which is bottom-up constructed from scratch, thereby generating only a small number of rules, whereas in [16] the initial number of rules is equal to the number of data tuples, which may be too large for large datasets. Secondly, our FNN is able to eliminate irrelevant inputs, rules, and rule conditions, which is necessary to effectively process a large amount of data. X. Z. Wang *et al* [16] used a confidence factor CF, representing

reliability of of the rules, defined as

$$CF = \mu_{T_{C1}} \mu_{F_{C1}} \mu_{T_{H11}} / ER \quad ,$$

where $\mu_{T_{C1}}$, $\mu_{F_{C1}}$, $\mu_{T_{H11}}$ are membership values for variables T_{C1}, F_{C1}, T_{H11}, respectively, and ER is the error. Then a $\lambda - Cut$ value was defined as the threshold for the rule to be worth keeping. In such way the rule base size can be reduced. However, in the example given with a $\lambda - Cut \times 10 = 6.0$, the reduced rule base still has multiple instances of the same rule, which makes the rule base ambiguous. For example, rules 2 and 10 are exactly the same,

$$\text{IF } F_{C1} \text{ is } Normal \ (\mu = 0.61) \text{ and } T_{C1} \text{ is } Normal \ (\mu = 1.00)$$

$$\text{THEN } T_{H11} \text{ is } Normal \ (\mu = 0.67) \quad ,$$

and rules 40 and 47 are different only in membership values, i.e., the rule 40 is

$$\text{IF } F_{C1} \text{ is } High \ (\mu = 0.67) \text{ and } T_{C1} \text{ is } High \ (\mu = 0.19)$$

$$\text{THEN } T_{H11} \text{ is } Normal \ (\mu = 0.55) \quad ,$$

and the rule 47 is

$$\text{IF } F_{C1} \text{ is } High \ (\mu = 1.00) \text{ and } T_{C1} \text{ is } High \ (\mu = 1.00)$$

$$\text{THEN } T_{H11} \text{ is } Normal \ (\mu = 0.22) \quad .$$

Consequently, it is difficult to see which rule is applicable to which case.

The FNN of Khan *et al* [8] is partially recurrent, i.e. the recurrent link is between the hidden and the input layer. As the output of the hidden layer and the network are equivalent only for the case where the network has one hidden node and one output node, the recurrency (current inputs depend on past outputs) has ambiguous meaning. Our FNN is fully recurrent, i.e., the recurrent link is from the output to the input layer, which results in the clear interpretation of the rule base. In addition, the network of Khan *et al* [8] is not self-constructing and it does not have a facility for elimination of irrelevant inputs, rules, and rule conditions, which make it unsuitable for large scale problems.

2 Fuzzy Neural Network Structure and Learning Algorithm

The structure of the proposed FNN is shown in Fig. 1. The network consists of four layers, i.e., the input layer, the input membership functions layer, the rule layer, and the output layer. The input nodes represent input variables consisting of the current inputs and the previous outputs. This recurrent structure provides the possibility to include temporal information, i.e., the network learns dynamic input-output mapping instead of static mapping as is feedforward neural networks [9]. It also speeds up the convergence of the network. In databases, data fields are either numerical or categorical. The input membership functions layer

generates input membership functions for numerical inputs, i.e., numerical data values are converted to categorical values. For example, the numerical values of *age* are converted to categorical values of *young*, *middle-aged*, and *old*. Each input node is connected to all membership function nodes for this input. We used piecewise linear-triangular membership functions for computational efficiency. The leftmost and rightmost membership functions are shouldered. Rule nodes are connected to all input membership function nodes and output nodes for this rule. Each rule node performs product of its inputs. The input membership functions act as fuzzy weights between the input layer and the rule layer. Links between the rule layer, the output layer and the input membership functions are adaptive during learning. In the output layer each node receives inputs from all rule nodes connected to this output node and produces the actual output of the system.

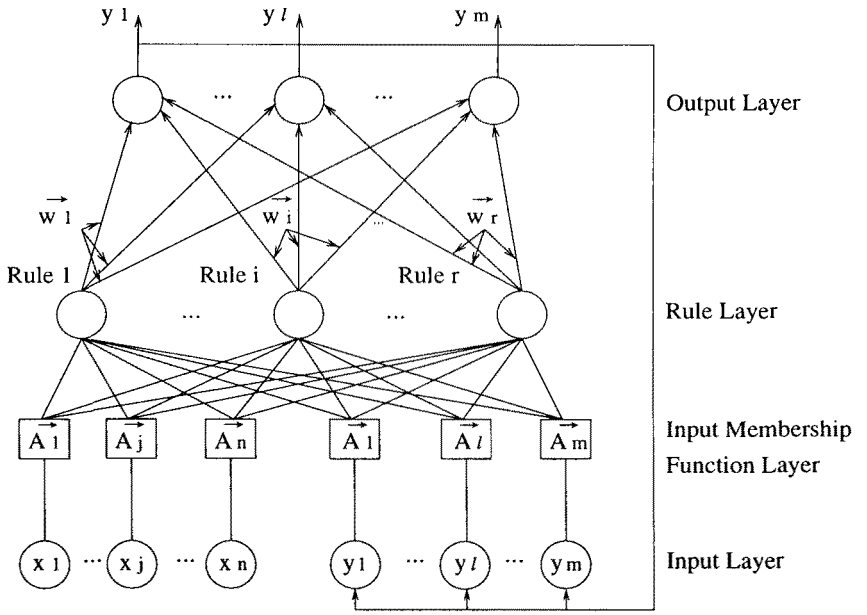

Fig. 1. The structure of our fuzzy neural network

The FNN structure-generation and learning algorithm is as follows:

0. Start with the number of input nodes equal to the sum of the input variables consisting of the current inputs and the past outputs $(n+m)$, and the number of output nodes equal to the number of output variables (m). The rule layer is empty, i.e., there are initially no rules in the rule base;
1. Add two equally spaced input membership functions along the operating range of each input variable. In such a way these membership functions will

satisfy $\epsilon - completness$, which means that for a given value of x of one of the inputs in the operating range, we can always find a linguistic label A such that $\mu_A(x) \geq \epsilon$. If the $\epsilon - completness$ is not satisfied, there may be no rule applicable for a new data input;

2. Create initial rule base layer using the following form for rule i:

$$\begin{aligned} &\text{Rule } i: \quad \text{IF } x_1 \text{ is } A_1^i \text{ and } \ldots x_n \text{ is } A_n^i \\ &\text{and } y_1(k-1) \text{ is } A_1^i \text{ and } \ldots y_m(k-r) \text{ is } A_m^i \\ &\qquad \text{THEN } y_1 = w_1^i \ , \ \ldots \ , \ y_m = w_m^i \ , \end{aligned} \tag{1}$$

where x_j and y_l $(l = 1, 2, .., m)$ are the current inputs and the past outputs, respectively, w_l^i is a real number. Here A_j^i is the membership function of the antecedent part of the i rule for the j input node, k is the time, and r is the delay. The membership value μ_i of the premise of the ith rule is calculated as fuzzy AND using the product operator

$$\mu_i = A_1^i(x_1) \cdot A_2^i(x_2) \cdot \ldots \cdot A_n^i(x_n) \ . \tag{2}$$

The output y_l of the fuzzy inference is obtained using weighted average

$$y_l = \frac{\sum_i \mu_i w_l^i}{\sum_i \mu_i} \ ; \tag{3}$$

3. Train the network using the following learning rules

$$w_l^i(k+1) = w_l^i(k) - \eta \frac{\partial \varepsilon_l}{\partial w_l^i} \ . \tag{4}$$

$$A_j^i(k+1) = A_j^i(k) - \eta \frac{\partial \varepsilon_l}{\partial A_j^i} \ , \tag{5}$$

where η is the learning rate. The objective is to minimize an error function

$$\varepsilon_l = \frac{1}{2}(y_l - y_{dl})^2 \ , \tag{6}$$

where y_l is the current output, y_{dl} is the target output. The learning rate η is adaptive to improve the speed of convergence, as well as the learning performance (accuracy). We start with a basic learning rate to enhance the learning speed. Whenever ε_l changes its sign, the learning rate is reduced according to the following iterative formula

$$\eta_{\text{new}} = rc \ \eta_{\text{old}} \ , \tag{7}$$

where rc is a coefficient in the range $(0, 1)$. The learning error ε_l asymptotically approaches zero or a pre-specified small value > 0 as the iteration number k increases;

4. If the degree of overlapping of membership functions is greater than a threshold (e.g. 0.9), combine those membership functions. We use the following *fuzzy similarity measure* [4]

$$Degree(A_1 = A_2) = E(A_1, A_2) = \frac{M(A_1 \cap A_2)}{M(A_1 \cup A_2)} \ , \tag{8}$$

where $\cap$ and $\cup$ denote the intersection and the union of two fuzzy sets A_1 and A_2, respectively. $M(\cdot)$ is the size of a fuzzy set, and $0 \leq E(A_1, A_2) \leq 1$. If an input variable ends up with only one membership function, which means that this input is irrelevant, delete the input. We can thus eliminate irrelevant inputs and reduce the size of the rule base. If the classification accuracy of FNN is below the requirement (e.g. 99%), and the number of rules is less than the specified maximum, go to step 6. Otherwise, go to step 5;
5. The generated rules are evaluated for accuracy and generality. We use a weighting parameter between accuracy and generality, the rule applicability coefficient (weighting of the rules) (WR) which is defined as the product of the number of the rule activations RA by the accuracy of the rule A in terms of misclassifications. All rules whose rule applicability coefficient WR falls below the defined threshold (e.g. 10) are deleted. Elimination of rule nodes is *rule by rule*, i.e., when a rule node is deleted, associated input membership nodes and links are deleted as well. By varying WR threshold a user is able to specify the degree of rule base compactness. The size of the rule base can thus be kept minimal. If the classification accuracy of the FNN after pruning is below the requirement (e.g. 90%), go to step 6, otherwise stop;
6. Add additional membership function for each input at its value at the point of the maximum output misclassification error. One vertex of additional membership function is placed at the value at the point of the maximum output error and has membership value unity; the other two vertices lie at the centers of the two neighbouring regions, respectively, and have membership values zero. As the output of the network is not a binary 0 or 1, but a continuous function in the range from 0 to 1, by firstly eliminating the errors whose deviation from the target values is the greatest, we can speed up the convergence of the network substantially;
7. Update the rule base layer in the same way as in step 2;
8. Retrain the network with the updated rule base layer using the algorithm given in step 3;
9. Go to step 4.

3 Experimental Results

To test our algorithm we used ten classification problems of different complexity defined in [1] on synthetic database with nine attributes given in Table 1. Attributes *elevel*, *car* and *zipcode* are categorical, and all others are non-categorical. Functions 1 to 5 have predicates with one (function 1), two (functions 2 and 4), and three attribute values (functions 4, 5, and 6). Functions 7 to 9 are linear

functions and function 10 is a non-linear function of attribute values. Consult Agrawal *et al* [1] for detailed description of the database and the functions.

Table 1. Description of attributes adapted from [1]

Attribute	Description	Value
salary	salary	uniformly distributed from 20K to 150K
commission	commission	salary $\geq 75K$ => commission = 0 else
		uniformly distributed from 10k to 75K
age	age	uniformly distributed from 20 to 80
elevel	education level	uniformly chosen from 0 to 4
car	make of the car	uniformly chosen from 1 to 20
zipcode	zip code of the town	uniformly chosen from 9 available zipcodes
hvalue	value of the house	uniformly distributed from n50K to n150K
		where $n \in 0...9$ depends on zipcode
hyers	years house owned	uniformly distributed from 1 to 30
loan	total loan amount	uniformly distributed from 0 to 500K

Attribute values were randomly generated according to uniform distribution as in [1]. For each experiment we generated 1000 training and 1000 test data tuples. As the tuples were classified into two classes only (Groups A and B), a single output node was sufficient. Default class was Group B. The target output was 1 if the tuple belongs to Group A, and 0 otherwise. We used the random data generator with the same perturbation factor $p = 5\%$ as in [1] to model fuzzy boundaries between classes. For comparison with our approach we used decision trees algorithms C4.5 and C4.5rules [12] on the same data sets. We also compared our results with those of a pruned feedforward neural network (NeuroRule) of Lu *et al* reported in [9] for the same classification problems. Table 2 shows the accuracy on the test data set, the number of rules, and the average number of conditions per rule, for all three approaches, with two sets of FNN parameters. We report only the rules for class A for C4.5 to make the comparison more objective, as C4.5 generates rules for both A and B classes, while both our FNN and NeuroRule [9] generate rules only for class A, using class B as a default rule.

With the first set of parameters in Table 2 (indicated by FN1) and compared to NeuroRule, our FNN produces the rule bases of less complexity for all functions, except 7 and 9. The FNN gives better accuracy for functions 6, 7, and 9, the same for function 1, and worse for the rest. Compared to C.4.5, the FNN gives less complex rule bases for functions 2, 4, and 9, the same for functions 2, 4, and 7, and more complex for the rest. Our FNN gives better accuracy than C4.5 on function 8, 9, and 10, similar for function 1, 6, and 7 and worse for the rest. The results for functions 8 and 10 were not reported in [9] (indicated by "N/A" in Table 2) as these functions lead to highly skewed data.

Table 2. Accuracy rates on the test data set, the number of rules, and the average number of conditions per rule for NeuroRule (NR) (performance data taken from [9]), the C4.5, and the FNN (FN). FN1 uses the following parameters: the learning rate $\eta = 0.0001$, the coefficient for learning rate adaptation $rc = 0.9$, required classification accuracy for training 99%, required classification accuracy after pruning 90%, maximum number of rules = 20, degree of overlapping of membership functions = 0.9, weighting of the rules $WR = 10$, maximum number of iterations $k = 10$. FN2 uses the same parameters as FN1, except weighting of the rules $WR = 15$

Func.	Accuracy				No. of Rules				No. of Conditions			
	NR	C4.5	FN1	FN2	NR	C4.5	FN1	FN2	NR	C4.5	FN1	FN2
1	99.91	99.9	99.9	100	2.03	2	2	3	2.23	1	1	1
2	98.13	99.4	96.8	93.3	7.13	3	3	4	4.37	3.33	2.33	2
3	98.18	99.8	97.5	100	6.70	5	5	6	3.18	2	2	2
4	95.45	99.3	95.0	95.3	13.37	11	7	8	4.17	4.18	3.14	3.13
5	97.16	98.6	96.4	93.0	24.40	4	5	7	4.68	4.75	4.2	4
6	90.78	95.8	94.7	95.1	13.13	8	9	10	4.61	3.37	3	3.1
7	90.50	95.4	95.2	95.9	7.43	12	10	13	2.94	2.17	2.9	2.54
8	N/A	98.1	99.1	98.9	N/A	2	3	4	N/A	1	2.67	2.75
9	90.96	92.6	96.7	97.2	9.03	12	9	11	3.46	2.75	3.56	3.64
10	N/A	95.5	96.0	97.0	N/A	6	8	9	N/A	2.17	4.5	4.67

In Table 2 for FN1 we attempted to obtain the most compact rule base to allow for easy analysis of the rules for very large databases, and thus easy decision making in real-world situations. If accuracy is more important than compactness of the rule base, it is possible to use our FNN with more strict accuracy requirement, i.e., a higher threshold for pruning the rule base (WR), thereby producing more accurate rules at the expense of rule base complexity. The final decision regarding complexity versus accuracy of the rules is application specific. Table 2 shows the performance comparison with a weaker compactness constrain, i.e., a larger WR, for FN2. In this case the FNN offers similar or better accuracy compared to both C4.5 and NeuroRule, except for functions 2, 4, and 5. Comparing the performance of FN2 with FN1, we see that relaxing the compactness constrain (increasing WR) improves accuracy for functions 1, 3, 4, 6, 7, 9, and 10. For functions 2, 5, and 8, however, relaxing the compactness constrain actually results in lower accuracy, which means that caution is necessary in selecting appropriate values of the weighting of the rules WR. As all problems in the table are different, the different compactness constrain values for each function may be needed. We used the same WR value for all functions to make a balanced comparison.

4 Conclusion and Discussion

In this paper we presented a dynamically-constructed recurrent fuzzy neural network for knowledge discovery from databases. Experiments were conducted to test the proposed approach to a well defined set of data mining problems given by [1]. The results shows that the presented approach is able to achieve accuracy and compactness comparable to both feedforward neural networks and decision tree methods, with more compact rule base than feedforward neural network for most of datasets used, and for some compared to decision tree approach. The proposed method is also able to achieve a higher accuracy on some datasets compared to both feedforward neural network and decision tree approaches. Incremental on-line learning of the proposed approach requires less time compared to off-line training of feedforward neural network approaches [9] due to the local updating feature of fuzzy logic, i.e., our FNN only updates the rules applicable to a current data while feedforward neural network globally updates the network. The proposed method eliminates extraction of symbolic rules phase in both decision trees and feedforward neural networks approaches. Our FNN permits updating of the rules along with changes in database contents due to its recurrent structure, in contrast to the decision tree and the feedforward neural networks methods.

Another important feature of databases need to be considered. Data may contain a certain level of noise due to statistical fluctuations or human errors. Using neural network based approach such as [9] and the one proposed in this paper should also greatly reduce the problem of data noise, due to ability of neural networks to deal effectively with noisy data. Testing of the proposed FNN in this aspect is subject of a future work.

References

1. Agrawal R., Imielinski T., and Swami A.: Database Mining: A Performance Perspective. IEEE Trans. Knowledge and Data Engineering **5** (1993) 914–925
2. Breiman L., Friedman J. H., Olshen R. A., and Stone C. J.: Classification and Regression Trees: Wansworth International (1984)
3. Cercone N. and Tsuchiya M.: Special Issue on Learning and Discovery in Knowledge-based Databases. IEEE Trans. Knowledge and Data Engineering **5** (1993)
4. Dubois D. and Prade H.: A Unifying View of Comparison Indices in a Fuzzy Set Theoretic Framework. in Yager R. R. (ed.) Fuzzy Sets and Possibility Theory: Recent Developments: Pergamon NY (1982)
5. Frawley W. J., Piatetsky-Shapiro G., and Matheus C. J.: Knowledge Discovery in Databases: An Overview. In Piatetsky-Shapiro G. and Frawley W. J. (eds.): Knowledge discovery in databases: AAAI Press/MIT Press (1991) 1–27
6. Gallant S. I.: Connectionist Expert Systems. Communications of the ACM **32** (1988) 153–168
7. Kerber R.: Learning Classification Rules from Examples. Proc. 1991 AAAI Workshop on Knowledge Discovery in Databases: AAAI (1991)
8. Khan E. and Unal F.: Recurrent Fuzzy Logic Using Neural Networks. Proc. 1994 IEEE Nagoya World Wisepersons Workshop (1994) 48–55

9. Lu H., Setiono R., and Liu H.: Effective Data Mining Using Neural Networks. IEEE Trans. on Knowledge and Data Engineering **8** (1996) 957–961
10. Piatetsky-Shapiro G.: Special Issue on Knowledge Discovery in Databases - from Research to Applications. Int. J. of Intelligent Systems **5** (1995)
11. Quinlan J. R.: Induction of Decision Trees. Machine Learning **1** (1986) 81–106
12. Quinlan J. R.: C4.5:Programs for Machine Learning Morgan Kaufmann: San Mateo CA (1993)
13. Quinlan J. R.: Comparing Connectionist and Symbolic Learning Methods. In S. Hanson, G. Drastall, and R. Rivest (eds.): Computational Learning Theory and Natural Learning Systems: MIT Press **1** (1994) 445–456
14. Shavlik J. W., Mooney R. J., and Towell G. G.: Symbolic and Neural Learning Algorithms: An Experimental Comparison. Machine Learning **6** (1991) 111–143
15. Towell G. G. and Shavlik J. W.: Extracting Refined Rules From Knowledge-based Neural Networks. Machine Learning **13** (1993) 71–101
16. Wang X. Z., Chen B. H., Yang S. H., McGreavy C., Lu M. L.: Fuzzy Rule Generation From Data for Process Operational Decision Support. Computer and Chemical Engineering **21** (1997) 661–666
17. Wu X.: Knowledge Acquisition from Databases. Ablex Publishing: Norwood NJ (1995)