A Fuzzy Neural Network for Data Mining: Dealing with the Problem of Small Disjuncts

Yakov Frayman[1], Kai Ming Ting[1] and Lipo Wang[2]
[1]Deakin University, School of Computing and Mathematics,
662 Blackburn Road, Clayton, Victoria 3168, Australia
[2]Nanyang Technological University, School of Electrical and Electronic Engineering,
Block S2 Nanyang Avenue, Singapore 639798
E-mail: {yfraym, kmting}@deakin.edu.au, elpwang@ntu.edu.sg

Abstract

In today's information age, data mining, i.e., extracting useful patterns or relationships from vast amount of data, has become increasingly important. Decision trees are currently the most popular tools for data mining. Despite many advantages in this approach, some aspects require improvements. A notable problem is known as the problem of small disjuncts, where the induced rules that cover a small amount of training cases often have high error rates. The purpose of the present paper is to show that a dynamically constructed recurrent fuzzy neural network can deal effectively with this problem.

1. Introduction

Data mining is one of the most promising fields for application of intelligent information processing technologies. It aims to extract useful patterns and nontrivial relationships from large collections of data records [17]. Hence, the users can be provided with a powerful tool for exploiting vast amount of stored data.

The most popular approaches to data mining at present are decision trees generated through symbolic inductive algorithms [4, 9]. Decision trees such as C4.5 [9] use the maximum generality bias to achieve a high predictive accuracy on disjuncts that cover a large proportion of training instances (large disjuncts). However, the use of a maximum generality bias often results in high error rates on disjuncts that cover a small number of training instances (small disjuncts). The low accuracy in the later cases stem from the use by decision trees such as C4.5 some form of probability estimates (most commonly the relative frequency) to estimate the quality of inductive rules. The unreliable probability estimates from small number of training instances resulted in *the problem of small disjuncts*, where specific rules often produce high error rates [7, 13, 15, 16].

There exist some remedies for C4.5 to overcome the problem of small disjuncts. This includes modifying original Bayes-Laplace formula [10], and using a composite learner consisting of C4.5 and an instance-based learning method [1] for small disjuncts [13]. Another possibility is to assign different misclassification costs for different classes that in effect will change the original frequency of data. However, this is not a straightforward task, as for example, the instance-based learning method suffers from the similar problem, termed *the problem of atypicallity* [14]. Thus, the problems of small disjuncts and atypicallity can be seen as manifestations of an intrinsic problem in learning systems. In this paper, we will approach the problem of small disjuncts using a fuzzy neural network that does not use a probability estimate in its rule induction. Thus, it should not depend on a relative frequency of a particular class in the overall data set.

2. Fuzzy Neural Network

The structure of the fuzzy neural network (FNN) is shown in Figure 1. The network consists of four layers, i.e., the input, the input membership function, the rule, and the output layers. 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 the static mapping in feedforward neural networks.

Each input node is connected to all membership function nodes for this input. The input membership functions act as fuzzy weights between the input layer and the rule layer. Piecewise-linear triangular membership functions that correspond to second-order B-splines [5] are used. This type of membership functions is simple to implement and is computationally efficient. The leftmost and rightmost membership functions are shouldered to cover for the whole operating range of input.

Rule node i represents fuzzy rule ($i = 1, 2, ..., r$):

$$\text{IF } x_1 \text{ is } A^i_{x_1} \text{ AND } ... \; x_n \text{ is } A^i_{x_n}$$

$$\text{AND } y_1(k-1) \text{ is } A^i_{y_1} \text{ AND } ... \; y_m(k-1) \text{ is } A^i_{y_m}$$

$$\text{THEN } y_1 = w^i_1 \;, \; \cdots \;, \; y_m = w^i_m \; . \tag{1}$$

Here x_j ($j = 1, 2, .., n$), and y_l ($l = 1, 2, .., m$), are the inputs and the outputs, respectively. w^i_l is a real number. A^i_q ($q = x_1, x_2, .., x_n, y_1, y_2, .., y_m$) is the membership function of the antecedent part of rule i for node q in the input layer, and k is the time. Each rule node is

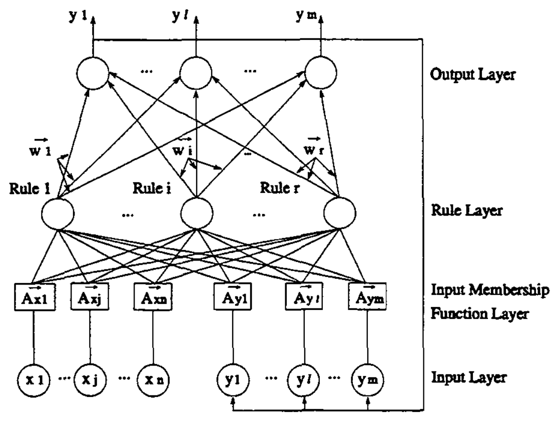

Fig. 1: The structure of the fuzzy neural network

connected to all input membership function nodes and output nodes for this rule. Links between the rule layer and the output layer, and the input membership functions are adaptive during learning. The membership value μ_i of the premise of the ith rule, is calculated as fuzzy AND using the product operator.

$$\mu_i = A^i_{x1}(x_1) \cdot A^i_{x2}(x_2) \cdot ... \cdot A^i_{xn}(x_n) \cdot A^i_{y1}(y_1) \cdot$$
$$\cdot A^i_{y2}(y_2) \cdot ... \cdot A^i_{ym}(y_m) \; . \tag{2}$$

Here μ_i indicates the degree to that the compound antecedent of the rule is satisfied.

The use of a product operator makes fuzzy inference to be fully differentiable at any point [8, 12]. On the other hand, the use of the truncation (MIN) operator would introduce derivative discontinuities both along the lines parallel to the input axes and along the main and minor diagonals [5]. The relational surface in this case would also have large areas where the system is not sensitive to any changes in input, and the output of fuzzy system is constant in these regions. Therefore, fuzzy inference that use truncation operators is not inherently robust rather the information lost during their operation produces undesirable fuzzy relational surface [5].

In addition, the use of the product operator for fuzzy AND produces a smooth output surface in contrast with the commonly used fuzzy MIN operator. The product operator forms multivariate membership functions. Such membership functions retain more information than when does the MIN operator is used. The latter scheme retains only one piece of information whereas the product operator combines n-pieces [5].

In the output layer, each node receives inputs from all rule nodes connected to this output node. The output y_l of the fuzzy inference is obtained using the weighted average (or center of gravity defuzzification)

$$y_l = \frac{\sum_i \mu_i w^i_l}{\sum_i \mu_i} \; . \tag{3}$$

The use of a weighted average (following the simplified fuzzy inference) produces a smoother output than the mean of maxima (MOM) defuzzification method and greatly reduces both the computational cost and the storage requirement of the algorithm [5].

The FNN structure generation and learning algorithms are as follows. Initially, $(n + m)$ nodes for the input layer and m nodes for the output layer are created. Here n and m are the number of the input variables (attributes) and the output variables (classes), respectively. Next, two equally spaced input membership functions are added along the operating range of each input variable. In such a way these membership functions satisfy ϵ-completeness, 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 the membership value $\mu_A(x) \geq \epsilon$. If the ϵ-completeness is not satisfied there may be no rule applicable for a new data input. The initial rule layer is created using equation (1).

The network is trained using the following general learning rule

$$y^i_l(k+1) = y^i_l(k) - \eta \frac{\partial \varepsilon_l}{dy^i_l} \; . \tag{4}$$

The learning rules for w^i_l and A^i_j are:

$$w^i_l(k+1) = w^i_l(k) - \eta \frac{\partial \varepsilon_l}{\partial w^i_l} \; , \tag{5}$$

for adaptation of the weights between the rule layer and the output layer, and

$$A_q^i(k+1) = A_q^i(k) - \eta \frac{\partial \varepsilon_l}{\partial A_q^i} \ . \qquad (6)$$

for adaptation of membership functions (fuzzy weights). Here η is the learning rate.

The objective is to minimize an error function

$$\varepsilon_l = \frac{1}{2}(y_l - y_{dl})^2 \ . \qquad (7)$$

Here y_l is the current output, and y_{dl} is the target output.

The learning rate η is variable: a relatively large learning rate to enhance the learning speed is used initially, i.e., $\eta = 0.01$. Whenever the error $\varepsilon(t)$ starts increasing, the learning rate is reduced according to the following iterative formula

$$\eta_{\text{new}} = r_c \ \eta_{\text{old}} \ . \qquad (8)$$

Here r_c is a coefficient in the range $(0, 1)$. Such decreasing learning rate algorithm can improve the speed of convergence, resulting in substantial reduction in training time, as well as in improvements in learning performance (accuracy).

The following recursive procedure is employed next. If the degree of overlapping of membership functions is greater than a specified threshold (e.g., 0.9), those membership functions are combined. The following *fuzzy similarity measure* [6] is used:

$$E(A_1, A_2) = \frac{M(A_1 \cap A_2)}{M(A_1 \cup A_2)} \ , \qquad (9)$$

Here $\cap$ and $\cup$ denote the intersection and 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. Irrelevant inputs thus can be eliminated and the size of the rule base can be reduced. Combining the membership functions is also done to eliminate the poor membership functions, and to replace them with the new ones that are likely to perform better.

If the classification accuracy is above or if the number of rule nodes is below the respective pre-specified threshold, the algorithm stops. Otherwise, add an additional membership function for all inputs at the point of the maximum output error. By firstly eliminating the errors whose deviation from the target values is the greatest, the output error can be reduced more efficiently and the convergence of the network can be substantially improved.

As the objective of data mining is to produce a small set of simple and accurate rules, there is a need to achieve a maximum compactness of the rule base while maintaining a high accuracy. The need of the compact rule base is twofold: it should allow for scaling of data mining algorithms to large dimensional problems, as well as providing the user with easily interpretable rules.

Consequently, a pruning phase of the algorithm was also implemented. The generated rules are evaluated for accuracy and generality. A weighting parameter between accuracy and generality, the rule applicability coefficient (weighting of the rules) (WR) is used. It is defined as the product of the number of the rule activation RA by the accuracy of the rule A. All rules whose rule applicability coefficient WR falls below a pre-defined threshold are deleted. When a rule node is deleted associated input membership functions and links are deleted as well. By varying the WR threshold the user is able to specify the degree of rule base compactness. In such a way, the size of the rule base can be kept minimal. Thus, effectively both construction and pruning phases are employed in the overall algorithm.

3. Experimental Results

The FNN method was tested on a car evaluation database from the Machine Learning Repository at University of California, Irvine [2]. The car evaluation database was derived from a simple hierarchical decision model [3, 18]. The reason for using this database is to exemplify the problem of small disjuncts when the data distribution is skewed.

The car evaluation database contains examples with the structural information removed, i.e., directly relates a car to six input attributes: buying, maint, doors, persons, lug-boot, and safety. The database has 1728 instances that completely cover the attribute space, 6 attributes and 4 classes. The attribute values are: buying = {v-high, high, med, low}, maint = {v-high, high, med, low}, doors = {2, 3, 4, 5-more}, persons = {2, 4, more}, lug-boot = {small, med, big}, and safety = {low, med, high}. The class distribution is as follows: class unacc = 1210 (70.0%), acc = 384 (22.2%), good = 69 (4.0%), vgood = 65 (3.8%). The data set was randomly split in two equal sets: training and testing, with the same distribution of classes in each set.

For comparison with the FNN approach, we used C4.5Rules [11] to make the results comparable with the rule-based FNN. C4.5Rules Release 8 with default parameters was used. The performance of the FNN and C4.5Rules on a car evaluation database is given in Table 1. The FNN gives higher accuracy that the C4.5Rules for the whole database as well as per each class, with a less complex rule base for class 'acc'. While both the FNN and C4.5Rules give quite similar accuracy for classes 'unacc' and 'acc', the FNN gives much higher accuracy on classes 'good' and 'vgood'. To note, the frequency for class 'good' in the database is 4.0%, and for class 'vgood' 3.8%.

Class	Accur. (%)		Rules		Cond.	
	C4	FN	C4	FN	C4	FN
unacc	98.0	98.4	9	10	20	24
acc	72.0	79.7	25	14	97	58
good	44.1	82.9	6	8	24	28
vgood	54.5	75.0	6	7	25	31
Overall	88.4	92.7	46	39	166	141

Table: 1: Accuracy on test data set, the number of rules, the number of conditions for C4.5Rules Release 8 (C4), and the FNN (FN) for the car evaluation database

4. Discussion and Conclusions

As can be seen from the results in Table 1, the FNN inductive learning (i.e., the error rate) is affected much less by the problem of small disjuncts, in contrast to C4.5Rules. This is because the FNN treats all classes with equal importance, whereas C4.5Rules gives preference to classes with a higher occurrence in the data set. On the other hand, C4.5Rules treats classes with lower support as less important (considering them as noise or exceptions). In reality, however, classes with low occurrence in the data set may not be the noise, but instead contain essential knowledge that one tries to extract from the database. For example, for the car evaluation database used in our experiments, the aim is not only to find which car is not suitable, but also to find which car is the most suitable. The FNN can offer much better advise to the user on finding cars in 'good' and 'vgood' classes, in comparison to C4.5Rules. Thus, the FNN approach to small disjuncts can be very useful in real-world situations, as the data of interest can often be only a small fraction of the available data.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[2] C. Blake, E. Keogh, and C. J. Merz, *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html], University of California, Department of Information and Computer Science: Irvine, CA, 1998.

[3] M. Bohanec, V. Rajkovic, "Expert system for decision making," *Sistemica*, vol. 1, no. 1, pp. 145-157, 1990.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wansworth International: Belmont, CA, 1984.

[5] M. Brown and C. Harris, *Neurofuzzy adaptive modelling and control*, Prentice-Hall, 1994.

[6] D. Dubois and H. Prade, "A unifying view of comparison indices in a fuzzy set theoretic framework," in R. R. Yager, (Ed.), *Fuzzy sets and possibility theory: recent developments*, Pergamon, NY, 1982.

[7] R. C. Holte, L. E. Acker, and B. W. Porter, "Concept learning and the problem of small disjuncts," *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, pp. 813-818, 1989.

[8] H. Nomura, I. Hayashi, and N. Wakami, "A learning method of fuzzy inference rules by descent method," *Proc. First IEEE Int. Conf. on Fuzzy Systems*, pp. 203-210, 1992.

[9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[10] J. R. Quinlan, "Improved estimates for the accuracy of small disjuncts", *Machine Learning*, vol. 6, no. 1, pp. 93-98, 1991.

[11] J. R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann: San Mateo, CA, 1993.

[12] I. Rojas, J. Ortega, F. J. Pelayo, and A. Prieto, "Statistical analysis of the main parameters in the fuzzy inference process," *Fuzzy Sets and Systems*, vol. 102, pp. 157-173, 1999.

[13] K. M. Ting, "The problem of small disjuncts: its remedy in decision trees," *Proc. 10th Canadian Conf. on Artificial Intelligence*, pp. 91-97, 1994.

[14] K. M. Ting, "The problem of atypicality in instance-based learning," *Proc. 3rd Pacific Rim Int. Conf. on Artificial Intelligence*, vol. 1, pp. 360-366, 1994.

[15] G. M. Weiss, "Learning with rare cases and small disjuncts," *Proc. 12th Int. Conf. on Machine Learning*, pp. 558-565, 1995.

[16] G. M. Weiss and H. Hirsh, "The problem with noise and small disjuncts," *Proc. 15th Int. Conf. on Machine Learning*, pp. 574-578, 1998.

[17] S. Weiss and N. Indurkhya, *Predictive data mining: a practical guide*, Morgan Kaufmann: San Francisco, CA, 1998.

[18] B. Zupan, M. Bohanec, I. Bratko, and J. Demsar "Machine learning by function decomposition," *Proc. 14th Int. Conf. on Machine Learning*, pp. 421-429, 1997.