ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 155-160

A New Rule Extraction Algorithm based on Interval Arithmetic[†]

† Carlos Hernández-Espinosa † Mercedes Fernández-Redondo †Mamen Ortiz-Gómez.

† Universidad Jaume I, Campus de Riu Sec, D. de Ingeniería y Ciencia de los Computadores., 12071 Castellón, Spain. e-mail: espinosa@inf.uji.es

Abstract. In this paper we propose a new algorithm for rule extraction from a trained Multilayer Feedforward network. The algorithm is based on an interval arithmetic network inversion for particular target outputs. The types of rules extracted are N-dimensional intervals in the input space. We have performed experiments with four database and the results are very interesting. One rule extracted by the algorithm can cover 86% of the neural network output and in other cases 64 rules cover 100% of the neural network output.

1. Introduction

Neural networks have been applied to a great number of applications and one of the most widely used neural network paradigms is Multilayer Feedforward. However, in same applications it is not only sufficient a correct classification of an input, it is also necessary an explanation of the classification [1]. One example is the medical diagnosis field, in this case, we need to provide a correct classification of the symptoms (the disease) and an explanation of the classification for the doctor.
A fundamental problem of neural networks is that the information they encode can not be easily understood by humans, for example, it is difficult to give an explanation on how they solve a particular problem.
One of the methods to solve the problem is rule extraction from a trained neural network. With this method, we tray to convert the information contained in a neural network in a set of rules that can be understood by a person.
There are many algorithm for rule extraction [2-5]. They differ in the type of rules extracted and many other characteristics. However, they lack from a common problem, the computational cost of the extraction of rules increases exponentially with the number of parameters in the neural network (weights or neurons).
In this paper, we propose a new algorithm for rule extraction from a trained Multilayer Feedforward network based on interval arithmetic. The algorithm is based in a network inversion for a particular target using the interval arithmetic properties. The type of rules extracted are N-dimensional intervals in the input space.
It has the problem of an exponentional computational cost increase with the number of inputs in the neural network, but other parameters like the number of weights or hidden units does not affect significantly the computational cost of the algorithm.

[†] This research work was supported by a Spanish CICYT project number TIC2000-1056.

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 155-160

The organization of the paper is the following. In section two we describe the neural network inversion method and the rule extraction algorithm. In section three we present the experimental results in four databases and finally the conclusions are in section four.

2. Theory

First, we will review the basic operations of interval arithmetic used in this paper. They are sum of intervals, multiplication of an interval by a number and the exponential [6].
Sum of intervals:

$$A+B=[a^L,a^U]+[b^L,b^U]=[a^L+b^L,a^U+b^U] \tag{1}$$

Where the superscripts L and U denote the lower and upper limits of the interval.
Product by a real number:

$$m\cdot A=m\cdot\left[a^L,a^U\right]=\begin{cases}\left[m\cdot a^L,m\cdot a^U\right] & if\ m\geq 0\\ \left[m\cdot a^U,m\cdot a^L\right] & if\ m<0\end{cases} \tag{2}$$

Exponential function:

$$\exp(A)=\exp\left(\left[a^L,a^U\right]\right)=[\exp(a^L),\exp(a^U)] \tag{3}$$

With these operations, we can calculate an interval output of the Multilayer Feedforward, for an interval input. The interval output of the hidden units are:

$$H_{P,j}=[H^L_{P,j},H^U_{P,j}]=f(Net_{P,j})=f([net^L_{P,j},net^U_{P,j}])=[f(net^L_{P,j}),f(net^U_{P,j})]$$

$$where\ \ Net_{P,j}=\sum_{i=1}^{Ninputs}w_{j,i}\cdot I_{P,i}+\theta_j \qquad Net_{P,i}=\left[net^L_{P,j},net^U_{P,j}\right]$$

$$where\ net^L_{P,j}=\sum_{i=1,w_{j,i}\geq 0}^{Ninputs}w_{j,i}\cdot I^L_{P,i}+\sum_{i=1,w_{j,i}<0}^{Ninputs}w_{j,i}\cdot I^U_{P,i}+\theta_j \tag{4}$$

$$net^U_{P,j}=\sum_{i=1,w_{j,i}\geq 0}^{Ninputs}w_{j,i}\cdot I^U_{P,i}+\sum_{i=1,w_{j,i}<0}^{Ninputs}w_{j,i}\cdot I^L_{P,i}+\theta_j$$

Where f is the standard sigmoid function and $I_{P,j}=[I_{P,j}^L,I_{P,j}^U]$ the input interval. Analogously for the interval outputs we have the equations 5.

$$O_{p,k}=[O^L_{P,k},O^U_{P,k}]=f(Net_{P,k})=f([net^L_{P,k},net^U_{P,k}])$$

$$where\ net^L_{P,k}=\sum_{j=1,w_{k,j}\geq 0}^{Nhidden}w_{k,j}\cdot H^L_{P,j}+\sum_{j=1,w_{k,j}<0}^{Nhidden}w_{k,j}\cdot H^U_{P,j}+\xi_k \tag{5}$$

$$and\quad net^U_{P,k}=\sum_{j=1,w_{k,j}\geq 0}^{Nhidden}w_{k,j}\cdot H^U_{P,j}+\sum_{j=1,w_{k,j}<0}^{Nhidden}w_{k,j}\cdot H^L_{P,j}+\xi_k$$

Now we will describe the interval arithmetic inversion, it is basically the same algorithm of neural network inversion [7], but in this case, the target will be an

interval vector and the error function will be the one of equation 6.

$$E_p = \frac{1}{4} \cdot \sum_{k=1}^{Noutput} \left\{ \left(t_{p,k}^U - o_{p,k}^U \right) + \left(t_{p,k}^L - o_{p,k}^L \right) \right\} \tag{6}$$

The inversion is accomplished by selecting and initial interval vector as the initial input $\{[i^L{}_1(0), i^U{}_1(0)], [i^L{}_2(0),i^U{}_2(0)], \ldots, [i^L{}_N(0),i^U{}_N(0)]\}$ and applying an iterative gradient descent algorithm similar to Backpropagation that will minimize the error value by changing the initial input. The equations are basically in 7.

$$i_{P,k}^L(n) = i_{P,k}^L(-1) - \eta \frac{\partial Error}{\partial i_{P,k}^L} \quad ; \quad i_{P,k}^U(n) = i_{P,k}^U(n-1) - \eta \frac{\partial Error}{\partial i_{P,k}^U} \tag{7}$$

The values of the partial derivates are in equation 8 and 9:

$$\frac{\partial Error}{\partial i_{P,k}^L} = -\frac{1}{2} \left\{ \sum_{k=1}^{Noutput} \left(t_{P,k}^L - o_{P,k}^L \right) o_{P,k}^L (1 - o_{P,k}^L) \cdot \left\{ \sum_i^{w_{k,i} \geq 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i.l} \right\} + \sum_{k=1}^{Noutput} (t_{P,k}^U - o_{P,k}^U) \cdot o_{P,k}^U \cdot (1 - o_{P,k}^U) \cdot \left\{ \sum_i^{w_{k,i} \geq 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} \right\} \right\} \tag{8}$$

$$\frac{\partial Error}{\partial i_{P,k}^U} = -\frac{1}{2} \left\{ \sum_{k=1}^{Noutput} \left(t_{P,k}^L - o_{P,k}^L \right) o_{P,k}^L (1 - o_{P,k}^L) \cdot \left\{ \sum_i^{w_{k,i} \geq 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i.l} \right\} + \sum_{k=1}^{Noutput} (t_{P,k}^U - o_{P,k}^U) \cdot o_{P,k}^U \cdot (1 - o_{P,k}^U) \cdot \left\{ \sum_i^{w_{k,i} \geq 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} \right\} \right\} \tag{9}$$

The type of rules we want to obtain are N-dimensional intervals in the input space like the following:

If $x_1 \subset [a^L{}_1,a^U{}_1]$, $x_2 \subset [a^L{}_2,a^U{}_2]$, …, $x_N \subset [a^L{}_N,a^U{}_N]$ then $\{x_1, x_2, \ldots, x_N\} \in$ Class K.

We should obtain the limits of the intervals a_i, b_i. They limit a N-dimensional interval in the input space and the whole N-dimensional interval has to be included in a classification class. An interval neural network inversion is used to get the intervals.

In order to obtain a rule, first, we will select a target value of the following type for one classification class (for example, class number 2): $\{[0,0.5]_{Class1}, [0.5,1.0]_{Class2}, [0,0.5]_{Class3}, \ldots, [0,0.5]_{ClassN}\}$. An output vector inside the above interval suppose a correct classification inside the class number 2, neuron number 2 is activated and the rest neurons are not.

Second, we will apply the inversion algorithm for the target. We expect that the initial input interval will evolve to give an interval whose output is inside the target interval selected, in this case, the final input interval will correspond to a valid rule.

We have performed simulations with three types of initial intervals in a two dimensional example. And the conclusions are that if the initial interval is a point

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 155-160

which corresponds to the correct classification of the initial point, during the inversion, the point will expand to an interval, the final limits of the interval will generally touch the borders of the classification class and the final results will normally correspond to a valid rule. We can see several examples in Fig. 1.

It is obvious that we can exploit this behavior of this type of initial intervals in order to propose an algorithm which can convert the information contained in the neural network into rules. The algorithm can be resumed as follows:

a) Select an initial point and calculate the output of the neural network for this input.
b) Select a target of the type described above, this target should agree with the classification class of the output of the neural network.
c) Apply the inversion algorithm and extract a rule.
d) Select a new point which is not included in the rules we have obtained before, and calculate the output of the neural network for this new point.
e) Select a target which agrees with the output of the neural network.
f) Apply the inversion algorithm and extract a new rule.
g) If we have not cover the whole input space go to step d).

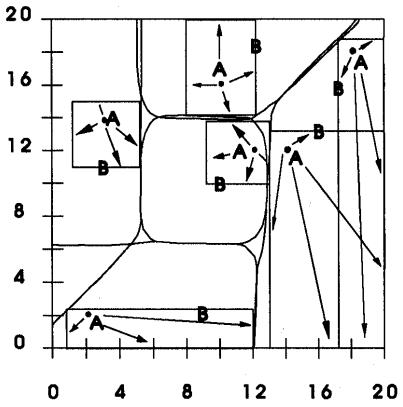

Fig.1 Example of point expansion by interval arithmetic inversion. A is the initial point, B is the final interval.

In order to test whether we have covered the input space and select a new initial point we can scan the input space with equally spaced points and test if the points are included in the rules. The space between the points will also influence in the final accuracy of the set of rules.

The problem of this scanning method is that its yields a computational complexity increase with the increase of the number of inputs and it can not be used with a high number of inputs. The methods of input or feature selection will play an important role to apply this algorithm.

There are other specific characteristics of this method. A rule will always have an output interval inside the target described above, so there will not be incorrect classification of points by the rule. Also another consequence is that there will not be overlapping among rules of different classes. And finally, the set of rules will not usually cover the total space of the neural network input, for example, the output {0.1, 0.8, 0.7} is not in the initial target intervals because two output units are activated and if an input has this output it will not be covered. We can say that the points where the classification of the neural network is not clear are not cover by the rules.

3. Experimental results

We have tested the neural network rule extraction algorithm with four database from

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 155-160

the UCI repository of machine learning databases. The databases are Balance Scale (BALANCE),), Liver Disorders (BUPA) and two of the Monk's Problems (MONK1 and MONK 2). We have selected this databases because the input dimensionality is at most six (http://www.ics.uci.edu/~mlearn/MLRepository.html).

We have applied the rule extraction algorithm to ten networks for each database which were trained with different random initialization of weights and different partition of data among training cross-validation and test.

In a rule extraction algorithm of this type we think that the two most important criterion for comparison are the fidelity of the rules and the number of rules extracted. By fidelity of rules we should understand how the results reproduced the behavior of the neural network.

In table 1 we have the results for the four databases.

Table 1. Results of the rule extraction algorithm.

Database	Prec. Space	Percen-tage	Not Cover	Total Percentage	Total Not Cover	Number of Rules (Nrule).	Nrule minimum	Percentage Nrule minimum
BALANCE	0.2	74.15	25.84	65.57	34.42	331.3	323	74.96
BALANCE	0.13	83.14	16.86	73.75	26.25	745.9	671	82.04
BUPA	0.2	92.59	7.41	92.59	7.41	6357.1	64	100
BUPA	0.13	85.11	14.89	85.11	14.89	19289.5	1	89.27
MONK1	0.2	81.10	18.90	81.10	18.90	8899.5	5036	82.44
MONK1	0.13	81.31	18.69	81.31	18.69	47211.4	35842	82.56
MONK2	0.2	93.21	6.79	93.21	6.79	7288	64	100
MONK2	0.13	83.16	16.84	82.69	17.31	12943.1	1	87.57

The second column (Prec. Space) is the distance in the input space between the points generated to construct the rules. A lower number means a higher number of points and therefore a higher number of rules.

We have randomly generate 10.000 point inside the input space with the condition that only one output unit is activated. The third column (Percentage) is the percentage of point covered by the rules and the fourth column (Not Cover) the percentage of points which were not cover.

Columns five and six are two percentages, in this case we have generated randomly 10.000 points without restriction inside the input space. The fifth column (Total Percentage) is the percentage of points cover by the rules and the sixth column (Total Not Cover) is the percentage of points not covered by the rules.

The seventh column (Number of Rules Nrule) is the mean number of rules generated by the algorithm. Column number eight is the minimum number of rules generated for a network and column number nine is the percentage of cover of this minimum number of rules.

We can see that, in general, the number of rules increases with the precision of the scanning space as it was expected. We can see very interesting results in the minimum number of rules, for example for the databases BUPA and MONK2 64 rules are enough to completely cover the input space. Also, in BUPA and MONK2 one rule cover more that 85% of the input space.

As we commented before, the rules will only cover the input space where only one output unit is activated and the rest are not. We can evaluate the maximum percentage

of point not covered, by subtracting the columns "Percentage" and "Total Percentage", they are 8.58% for the database BALANCE, 0.% for BUPA, 0% for MONK1 and 0.47% for MONK2. As we can see this effect is not so important in the experimental results.
The results of mean percentage of covering are in general good. But if we want to increase the percentage of covering we can generate random points outside the covering of the rules and extract new rules from this points. In a rule the initial point is usually covered by the rule (see Fig. 1), therefore the new rule will usually cover part of the input space not cover by the rest of the rules (the initial point is not contained in any rule).
We have applied this technique with the databases BALANCE which got the lower percentage of covering and the results are in Table 2.

Table 2. Results of the rule extraction algorithm.

Database	Number of Points	Percen-tage	Not Cover	Total Percentage	Total Not Cover	Number of Rules (Nrule).	Nrule minimum	Percentage Nrule minimum
BALANCE	5.000	95.19	4.81	85.45	14.54	2963	2706	95.87

4. Conclusions

We have presented a new algorithm for rule extraction from a trained Multilayer Feedforward neural network. The algorithm is based on an interval arithmetic network inversion for particular interval target outputs. The type of rules extracted are N-dimensional intervals in the input space. The experimental results are encouraging, one rule extracted by the algorithm can cover 86% of the input space of the neural network, and in other cases 64 rules cover 100% of the neural network.

References

1. A. Maren, C. Harston y R. Pap, Handbook of Neural Computing Applications, Academic Press Inc., 1990.
2. Lu, H., Setiono, R., Liu, H., "Effective Data Mining Using Neural Networks", IEEE Trans. on Knowledge and Data Engineering, vol. 8, no. 6, pp.957-961, 1996.
3. Thrun, S., "Extracting Rules from Artificial Neural Networks with Distributed Representations", Advances in Neural Information Processing Systems 7, pp. 505-512, 1995.
4. Gupta, A., Lam, S.M., "Generalized Analytic Rule Extraction for Feedforward Neural Networks", IEEE Trans. on Knowledge and Data Engineering, vol. 11, no. 6, pp. 985-991, 1999.
5. Narazaki, H., Shigaki, I., Watanabe, T., "A Method for Extracting Approximate Rules from Neural Network", Proc. of the IEEE Int. Conf. on Fuzzy Systems, vol. 4, pp. 1865-1870, 1995.
6. Alefeld, G., Herzberger, J., Introduction to Interval Computations, Academic Press, New York, 1983.
7. Linden, A. and Kinderman, J., "Inversion of Multilayer Nets", in Proc. of the Int. Conf. on Neural Networks, Washington D.C., vol. 2, pp. 425-30, 1989.