Using a Neural Network and Genetic Algorithm to Extract Decision Rules

Karen Blackmore
School of IT,
Charles Sturt University
Bathurst, NSW, Australia
kblackmore@csu.edu.au

Terry Bossomaier
School of IT,
Charles Sturt University
Bathurst, NSW, Australia
tbossomaier@csu.edu.au

Abstract

Rule extraction from neural networks often focusses on exact equivalence and is often tested on relatively small canonical examples. We apply genetic algorithms to the extract approximate rules from neural networks. The method is robust and works with large networks. We compare the results with rules obtained using state of the art decision tree methods and achieve superior performance to straight forward application of the WEKA implementation of the C5 algorithm, J48.PART.

Keywords

Neural networks, rule extraction, genetic algorithms, rule based classifiers.

INTRODUCTION

It seems somewhat strange to extract rules from artificial neural networks (ANN), when neural solutions were thought to be distinctive. The huge growth of activity in neural networks from the mid-80s occurred because they seemed to be able to solve problems intractable by conventional logic-based artificial intelligence. In fact a series of seminal papers [1], Abu Mostafa asserted that neural networks were particularly effective on problems of high algorithmic complexity [12], common in many pattern recognition. Such problems intrinsically require very many complex rules for their solution.

However, the success of ANNs in many areas led to a deeper understanding of how they work and detailed comparisons with statistical and other methods, such as the classic book by Ripley [16]. But the ease of use, resilience to lack of understanding of search space, and the ready availability of good software ensures the continuing use of ANN methods.

But in some situations where an ANN solution is to be distributed to a large client base, with little background in ANNs, for extended use in safety critical applications, some intuitive support is highly desirable. In such situations a set of rules is very valuable because of its greater ease of comprehension and its link to common sense and experience.

In earlier work [5] we showed that neural networks outperform state of the art decision tree methods on a human classification problem. In the present paper we apply genetic algorithms to extract rules from the network which themselves are superior to those we obtained from decision tree analysis.

BACKGROUND TO RULE EXTRACTION

A wide variety of methods are now available, recently reviewed by Andrews et al. [2]. Tickle et al [20] revisits the Andrews classification of rule extraction methods and emphasise distinction between decompositional and pedagogical approaches. Rule extraction methods usually start by finding a minimal network, in terms of number of hidden units and overall connectivity. Setiono [18] for example adds penalty terms to the error function to bias back-propagation like training towards such sparse networks. The next simplification, the key feature of the method, is to quantize or cluster the hidden unit activations. It is then possible, link by link, to extract combinations of inputs which will activate each hidden unit, singly or together and thus output generate rules. This unit by unit analysis characterises the decompositional approach. Although it can yield exact representations, the computational time may grow exponential with number of inputs (attributes) as noted by Tickle et al. [2] for decompositional algorithms such as Subset and KT.

Taha and Ghosh [19] suggest for binary inputs such as our data generating a truth table from the inputs and simplying the resultant Boolean function. But this simplification is itself combinatorially nasty and thus the method works only for small networks. They also refine the Liu and Setiono [13] methods using linear programming. The test networks have just a few inputs.

The growth of computational time with number of attributes makes minimising the size of the neural network essential and some methods evolve minimal topologies. Santos et al [17] use genetic algorithms in combination with the Setiono rule extraction to optimise network topology using quality indices on the rules extracted as a fitness function to guide network evolution.

The pedagogical approaches treat the neural network as a black box [20] and use the neural network only to generate test data for the rule generation algorithm. Keedwell et al.[11] use genetic algorithms to evolve rules directly. They use a wild-card like representation where each input to the

neural network is represented explicitly in the chromosome by one or more bits with zeros for don't care inputs. Our method is broadly similar, but since the number of inputs we have is very high, we use a different encoding scheme as discussed below. The Keedwell approach requires a specific term in the fitness function to make the rules as small as possible, since each chromosome admits every possible attribute. In our approach, we enforce maximum size predicates and number of rules thus reducing the size of the search space.

METHODOLOGY

The following methodology was employed:

1. Format dataset for training.
2. Generate random splits of the dataset for cross-validation.
3. Perform classification using rule based classifier.
4. Train the neural network
5. Extract rules from neural network using a genetic algorithm
6. Assess and compare the predictive accuracy of the rules from the neural network with those obtained using the rule based classifier.

Formatting of the Dataset

The data for analysis was provided by Foy as part of a larger study [8]. The data set contained 24 mulit-value categorical input and 3 output variables for 357 human profiling cases (see [4,5] for a detailed discussion and description). For classification using *J48.PART*, the data was converted to *ARFF* format [21], which provides the attribute definitions and the data instances without specifying the attribute for prediction. Selection of the prediction attribute and filtering of any unwanted attributes is carried out within the Weka scheme.

For neural network training, the dataset was converted to numeric binary values. This resulted in 96 inputs and 3 outputs due to the conversion of the multi-value categorical inputs.

Random Splits for Cross-Validation

Evaluation of an algorithm's predictive ability is best carried out by testing on data not used to derive rules [21], thus all training was carried out using tenfold cross-validation. Cross-validation is a standard method for obtaining an error rate of the learning scheme on the data [21]. Tenfold cross-validation splits the data into a number of blocks equal to the chosen number of folds (in this case 10). Each block contains approximately the same number of cases and the same distribution of classes. Each case in the data is used just once as a test case and the error rate of the classifier produced from all the cases is estimated as the ratio of the total number of errors on the holdout cases to the total number of cases. Overall error of the classification is then reported as an average of the error obtained during the test phase of each fold.

Variations in results for each iteration in a cross-validation occur depending on the cases used in the training and hold-out folds, which can lead differences in overall results. Thus, rather than use the cross-validation function, which is part of the rule based classifier, the cases in the dataset were randomized and split into ten separate train and test sets. Using this method, both the rule based classifier and the neural network were trained using the exact same fold splits.

Rule Based Classifier

Algorithms that derive rule sets from decision trees first generate the tree, then transform it into a simplified set of rules [9]. Based on results from previous research [3], the *WEKA J48.PART* [21] algorithm was selected to derive and evaluate rules sets from the training data.

Neural Network Training

The ANN was feed-forward, and one hidden layer was found to be sufficient [5], with 6 hidden units. Training was carried out using the neural network toolbox from Matlab [14], with the Levenberg-Marquardt algorithm and log-sigmoid transfer function [16].

Genetic Algorithm

Genetic algorithms [15], [10] are now a mature optimisation technique, achieving good results on NP hard problems in practical times. We employed a simple genetic algorithm with a single population, no self-adaptation of parameters, and a variety of conventional cross-over mechanisms.

The genetic code is of fixed length and encodes a maximum number of rules and conditions per rule. Binary represented integers are used to describe each condition in the rule with a further bit setting this condition to true or false. The networks have 90 inputs, so 7 bit integers are used for each predicate. 38 values are thus unused, but they serve an important function. If input values appear in this high range, they do not generate a condition. Thus, the genetic code can describe rules with fewer conditions in the predicate than the maximum allowable.

After the predicate comes the rule outcome. Together this chunk of predicate and outcome is repeated to the maximum number of rules allowed. Since the data has three outcomes, a fourth outcome (from 2 bits) is available. This could be used to indicate a null rule, but in the work presented herein, has a different function. The neural networks seem to generalise well and were tested with extensive cross-validation [5]. However, if we now generate random inputs and observe the neural network output, we can sometimes get inconsistent outputs. Thus there are regions of the search space, which would not occur in practice, in which

the neural network is less secure. This fourth bit encodes the ``don't know'' case.

As an illustration, suppose we have two rules each with three conditions. The chromosome will have 136 bits as shown in Figure 1. Square brackets denote the number of bits in each gene.

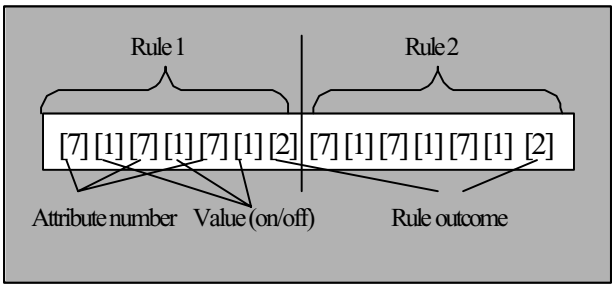

Figure 1. Example chromosome for a two rule, three condition set.

The fitness function is simply the aggregate performance of the rule set measured against the neural network outputs. No majority voting on rules is imposed, so rules giving erroneous classification feed through to the fitness function. The approach is holistic. No one rule, regardless of number of cases it classifies, is given explicit preference over any other. This holistic approach avoids the brittleness of other methods, since the rules are optimised as a set rather than determined in sequence according to some criterion.

The genetic algorithm methods were implemented using the Genetic Algorithm Toolbox [6]for Matlab [14].

RESULTS

The predictive accuracy of the rule based classifier and the ANN, using both random and fixed initial weights and biases, were compared. As can be seen in both Table 1 and Figure 2, the ANN, regardless of starting conditions, achieved superior accuracy over the rule based classifier (WEKA). These results are consistent with previous research [5].

The rule based classifier achieved 70% predictive accuracy, whereas the ANN was able to, on average, correctly predict outcomes 99% of the time. Additionally, as each fold contained exactly the same train and test cases for all methods, it can be seen that classification errors vary randomly, with high error rates not linked to specific folds For example, the Weka method had the highest error on the sixth fold, whereas the ANN with random starting conditions had the highest error on the fourth fold. From the results of this ten-fold cross validation, the ANN with fixed starting conditions would appear to be most stable.

Table 1. Error Comparison

Fold	Weka	NN_Random	NN_Fixed
1	31.4	0.00	0.84
2	31.4	1.40	1.40
3	14.3	0.00	0.56
4	30.5	5.04	2.24
5	36.1	1.68	1.40
6	41.7	2.24	2.24
7	19.4	0.56	1.12
8	22.2	0.56	0.56
9	38.9	0.84	1.40
10	33.3	1.68	2.52
Avg Err	29.9	1.40	1.43

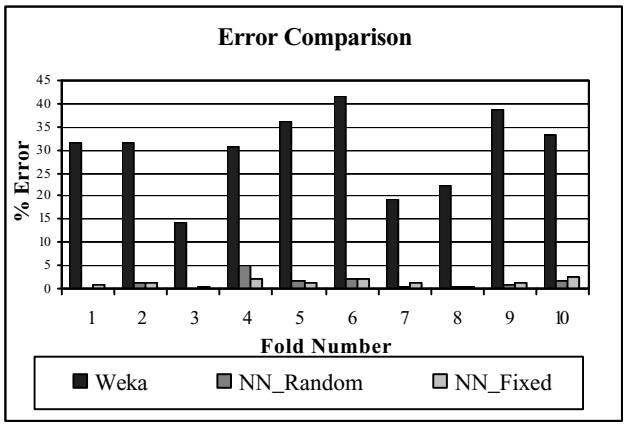

Figure 2. Graph Comparing Errors

Genetic Algorithm Parameters

The genetic algorithm initialized with the following parameters:

i. Number of individuals in the population: 40

ii. Maximum number of generations: 1000

iii. Generation gap: 1

iv. Selection function: 'sus' (stochastic universal sampling)

v. Maximum number of conditions: 3

vi. Number of rules: 18

Variations in the values of the initialization parameters were used, however, best results were achieved with the values above. The genetic algorithm converged quickly within the first 100 generations, with minimal improvement in fit in the following 900 generations (Figure 3).

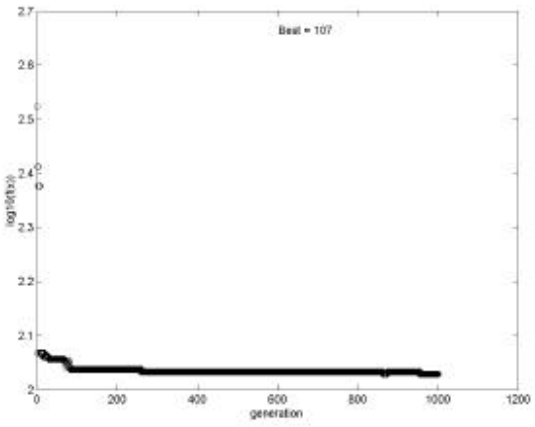

Figure 3. Convergence over 1000 generations

Rule set from the Genetic Code

The following ruleset was derived from the neural network using the genetic algorithm.

1. IF behavioural = Condition3, on, AND region = Condition 1, off, AND Age = Condition3, on, THEN =outcome1
2. IF mn_problems = Condition3, off, AND reporter = Condition1, on, AND Marital_status = Condition2, on, THEN =outcome1
3. IF longterm_stressors = Condition2, on, THEN =outcome2
4. IF time = Condition2, on, AND risk = Condition8, on, THEN =outcome2
5. IF Age = Condition3, on, THEN =outcome2
6. IF reporter = Condition2, off, AND dependen = Condition2, off, AND time = Condition4, off, THEN =outcome4
7. IF occupation = Condition2, off, THEN =outcome1
8. IF time = Condition4, off, AND last_seen = Condition2, off, AND day = Condition4, off, THEN =outcome1
9. IF day = Condition1, off, THEN =outcome2
10. IF mn_problems = Condition6, on, AND longterm_stressors = Condition4, off, AND day = Condition3, on, THEN =outcome2
11. IF risk = Condition4, off, AND character = Condition1, off, THEN =outcome1
12. IF Age = Condition2, on, THEN =outcome2
13. IF season = Condition3, off, AND day = Condition1, on, THEN =outcome2
14. IF occupation = Condition6, on, AND longterm_stressors = Condition2, off, AND reporter = Condition1, off, THEN =outcome3
15. IF mn_problems = Condition2, on, AND longterm_stressors = Condition1, off, THEN =outcome3
16. IF occupation = Condition2, off, AND occupation = Condition7, on, THEN =outcome2
17. IF urban = Condition2, off, AND history = Condition2, on, AND season = Condition1, off, THEN =outcome2
18. IF appearance = Condition1, off, AND ph_outcome2 = Condition2, on, AND reporter = Condition4, on, THEN =outcome3

Although the maximum number of conditions per rule is set at three, as mentioned previously, the genetic code can describe rules with fewer conditions in the predicate than the maximum allowable. The ruleset generated by the genetic algorithm produced 8 rules with the maximum three clauses, 5 rules with two clauses and 5 rules with one clause (Figure 4). The rules were valid for 306 or 86% of cases in the dataset. This represents a significant improvement over the 70% accuracy achieved by the rule based classifier. Importantly, this improvement in accuracy was achieved using fewer rules (18 as apposed to 22 for the rule based classifier) with fewer clauses (see Figure 4), thus abiding by the minimum description length principle [7].

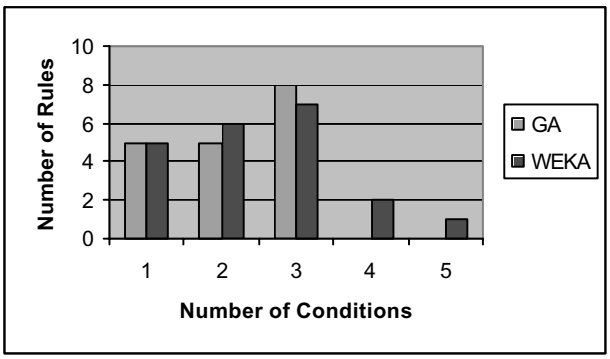

Figure 4. Comparison of Number of Rules and Conditions

DISCUSSION

Keedwell et al. [11] also find on manufactured test data with varying amounts of noise. They found a slight advantage for their GANN method when noise levels are high. Their technique uses additional heuristics after a set of rules has been generated by the genetic algorithm. The rules obtained may be contradictory and a voting system is used for choosing the classification in the event of conflict. Our fitness function explicitly penalises rules which generate incorrect outputs in some cases, thus the rule sets we evolve do not need further refinement or heuristics. Pedagogical methods of this kind may behave quite well with increasing size of neural network and number of attributes. The computation time for the neural network generation of test classifications grows approximately linearly with the number of connections, which is favourable. The complexity of the genetic algorithm depends on the size of the rule set, each iteration depending linearly on the number of rules and depends only weakly on the number of conditions in the current implementation. But, most significantly, it is most effective, when regardless of the size of neural network, a small number of rules capture the important characteristics of the

data, i.e. where the data is essentially of low algorithmic complexity [12]. It is in such cases that the rule extraction process will be of most value to the end user.

CONCLUSION

The genetic algorithm approach described here determines rules better than those found from various decision tree based methods. It is simple to implement, but requires considerable computing resources for the large neural networks of the present paper. As such its greatest value is for determining heuristic rules for medium term use by practitioners who welcome the intuitive description that rules provide.

REFERENCES

[1] Abu-Mostafa, Y. Complexity of random problems. In: *Complexity in Information Theory*, Anonymous Springer-Verlag, 1986.

[2] Andrews, R., Diederich, J., and Tickle, A. B., A survey and critique of techniques for extracting rules from trained artificial neural networks *Knowledge Based Systems*, vol. 8, pp. 373-389, 1995.

[3] Blackmore, K. and Bossomaier, T. R. J., "Comparison of *See5* and *J48.PART* Algorithms for Missing Persons Profiling," *ICITA2002,* Bathurst, 2002.

[4] Blackmore, K. L., Bossomaier, T. J. R., Foy, S., and Thomson, D., "Data mining of missing persons data," *1st International Conference on Fuzzy Systems and Knowledge Discovery,* Orchid Country Club, Singapore, 2002.

[5] Blackmore, K. L. and Bossomaier, T. R. J., "Soft Computing Methodologies for Mining Missing Person Data," *submitted to 2002 IEEE International Conference on Data Mining,* Maebashi TERRSA, Maebashi City, Japan, 2002c.

[6] Chipperfield, A. J., Fleming, P. J., Pohlheim, H., and Fonseca, C. M., "A genetic algorithm toolbox for MATLAB," *Proceedings of the International Conference on Systems Engineering,* Coventry, UK.

[7] Domingos, P. , The role of Occam's razor in knowledge discovery *Data Mining and Knowledge Discovery*, vol. 3, pp. 409-425, 1999.

[8] Foy, S. *PhD Thesis, forthcoming.*, (UnPub)

[9] Frank, E. and Witten, I. H., "Generating accurate rule sets without global optimization," *Machine Learning: Proceedings of the Fifteenth International Conference,* Madison, Wisconsin, pp. 144-151, 1998.

[10] Holland, J. H., Genetic algorithms *Scientific American*, vol. pp. pp. 45-50, Jul, 1992.

[11] Keedwell, E., Narayanan, A., and Savic, D., Creating rules from trained neural networks using genetic algorithms *International Journal of Computers, Systems and Signals (IJCSS)*, vol. 1, 2000.

[12] Li, M. and Vitanyi, P. *An Intorduction to Kolmogorov Complexity and its Applications*, 1997: Springer-Verlag,

[13] Liu, H. and Setiono, R., Incremental feature selection *Journal of Applied Intelligence*, vol. 9 , pp. 217-230, 1998.

[14] Mathworks Inc., Matlab Student Version , ver. 6.1 Release 12.1, rel. 2001. Mathworks Inc.

[15] Mitchell, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, MIT Press, 1996.

[16] Ripley, B. D. *Pattern Recognition and Neural Networks*, UK: Cambridge University Press, 2001.

[17] Santos, R., Nievola, J. C., and Freitas, A. A., "Extracting comprehensible rules from neural networks via genetic algorithms," *Proceedings of the 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN-2000),* San Antonio, TX, USA.

[18] Setiono, R., A penalty-function approach for pruning feedforward neural networks *Neural Computation*, vol. 9, pp. pp. 185-204, Jan, 1997.

[19] Taha, I. and Ghosh, J., Three techniques for extracting rules from feedforward networks *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 6, 1996.

[20] Tickle, A., Andrews, R., Golea, M., and Diederich, J., The truth is in there: directions and challenges in extracting rules from trained artificial neural networks *IEEE Transactions on Neural Networks*, vol. 9, pp. 1057-1068, 1998.

[21] Witten, I. and Frank, E. *Data mining: practical machine learning tools and techniques with Java implementations*, San Francisco: Morgan Kaufmann, 2000.