Optimising a Neural Tree Classifier Using a Genetic Algorithm

Wanida Pensuwon **Rod Adams** **Neil Davey**

W.Pensuwon@herts.ac.uk **R.G.Adams@herts.ac.uk** **N.Davey@herts.ac.uk**

Computer Science Dept. University of Hertfordshire, Hatfield, Herts, AL10 9AB,United Kingdom.,
Tel: +44 01707 284321

Abstract. *This paper documents experiments performed using a GA to optimise the parameters of a dynamic neural tree model. Two fitness functions were created from two selected clustering measures, and a population of genotypes, specifying parameters of the model were evolved. This process mirrors genomic evolution and ontogeny. It is shown that the evolved parameter values improved performance.*

1 INTRODUCTION

Organising the output nodes of a competitive network into a tree structure may produce potential improvements in both convergence and recall rates. It may also make the network more stable in responding to changes in the data being classified. However a limitation of most clustering algorithms is that they require the number of classes to be predefined. This is a problem if no a-priori knowledge about the data is available [3].

Dynamic neural tree networks (DNTN) may avoid the above limitation by dynamically creating nodes and automatically creating a tree structure. However, all the original models require parameters to be externally set, which will significantly influence the hierarchical structure and the classification that is produced [8, 11, 12].

A more recent DNTN is the Competitive Evolutionary Neural Tree (CENT), [1,2,3]. This model is comparatively robust, with respect to its parameter settings when compared with other DNTNs [2]. Nonetheless, it is unclear precisely how the internal parameters affect the performance, which lead to the investigation reported here. In order to explore the parameter space a good default set of parameters was needed, and a genetic algorithm (GA) was used as the search vehicle. A given genome (a set of parameters), together with a data set allowed CENT to produce a classificatory tree. The fitness of these trees was calculated using two clustering measures (described in section 4). Nine data sets were used.

2 ALGORITHM

In CENT, the tree structure is created dynamically in response to structure in the data set. The neural tree starts with a root node with its *tolerance* (the radius of its classificatory hypersphere) set to the standard deviation of input vectors and its position set to the mean of input vectors. It has 2 randomly positioned children. Each node has two counters, called *inner* and *outer*, which count the number of occasions that a classified input vector is within or outside tolerance, respectively. These counters are used to determine whether the tree should grow children or siblings once it has been determined that growth is to be allowed.

2.1 TOP-LEVEL ALGORITHM

At each input presentation, a recursive search through the tree is made for a winning branch of the tree. Each node on this branch is moved towards the input using the standard competitive neural network update rule.

Any winning node is allowed to grow if it satisfies 2 conditions. It should be mature (have existed for an epoch), and the number of times it has won compared to the number of times its parent has won needs to exceed a threshold. A finite limit is put on the number of times a node attempts growth.

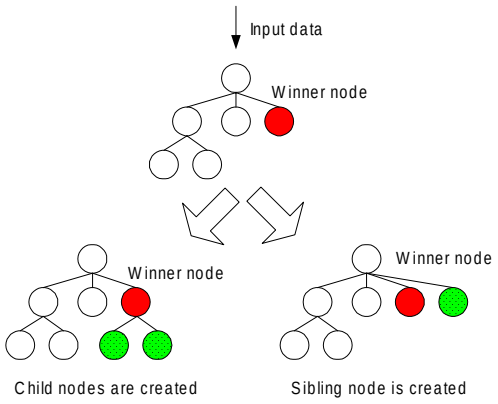

Figure 1. Process of growing a tree

When a node is allowed to grow, if it represents a dense cluster, then its inner counter will be greater than its outer counter and it creates two children. Otherwise, it produces a sibling node. The process of growing is illustrated in figure 1. To improve the tree two pruning algorithms, short and long term, are applied to delete the insufficiently useful nodes. The short-term pruning procedure deletes nodes early in their life if their existence is not improving the previous classificatory error according to an *ErrorAcceptance (EA)* parameter value. The long-term pruning procedure removes a leaf when its activity is not greater than an *epsilon* parameter. See figure 2 for a pruning process.

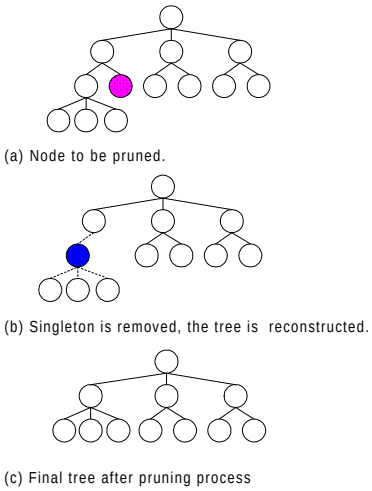

Figure 2. Pruning process

3 PARAMETER SETTINGS

The parameters that are the object of this investigation are briefly described in this section.

The *growthparameter* is the maximum times a node is allowed to grow. The *K* parameter influences how easy it is for a node to grow - its activity must satisfy:

$$\frac{Node's\ activity}{Parent's\ activity} > \frac{1}{Parent's\ number of\ child + K}$$

As *K* is increased, it makes node growth easier, and thereby bigger trees will result.

The *A* and *B* parameters determine the new tolerance (*tol*) of any children produced.

$$New\ children's\ tol = Parent's\ tol * C$$

$$where\ \ C = \left(A + \frac{Outer}{B * Inner} \right)$$

An increase in *A*, and/or a decrease in *B* tend to make the trees deeper.

When a node produces a sibling these two nodes have larger tolerance values, mediated by Sb_{tol}.

$$New\ Tol = Old\ Tol * (1 + Sb_{tol})$$

Increasing this parameter limits sibling growth.

EA is a parameter used in short-term pruning. The condition to determine when a node should be deleted is as below:

$$\frac{(Total\ sibling's\ sum\ of\ error) * (Number\ of\ sibling)}{Parent's\ previous\ error} > EA$$

In a long-term Pruning procedure, a leaf node is deleted when its activity is not greater than *epsilon*.

4 CLUSTER MEASURES

The general goal in many clustering applications is to arrive at clusters of objects that show small within-cluster variation relative to the between-cluster variation [7]. Clustering is difficult as many reasonable classifications may exist for a given data set, moreover it is easy for a clusterer to identify too few or too many clusters. Suitable cluster criterion measures are therefore needed [4].

An initial investigation, concentrated on 10 clustering criterion selected from a comparative evaluation of Milligan and Cooper [9] and another 2 hierarchical methods [5]. From this study, 2 measures were chosen: the *gamma measure*, which measures the flat partitioning performance and the *hierarchical distortion* that assesses a hierarchical clustering against the data.

5 GENETIC ALGORITHM OPTIMISATION

Searching a complex space of problem solutions often involves a balance between two apparently conflicting objectives. The first is exploiting the best solutions currently available and the second is exploring the space. GAs have been identified as a general purpose search strategy that strikes a reasonable balance between exploration and exploitation [10].

The seven parameters, described earlier, were encoded in a binary genome of 40 bits. A population of 50 individuals with random initialisation, were evolved using the GENESIS package from John Grefenstette [6], for 50 generations. The mutation rate was 0.001 and the crossover rate was 0.6, using dual point

crossover. Selection was rank based using an elitist strategy [10].

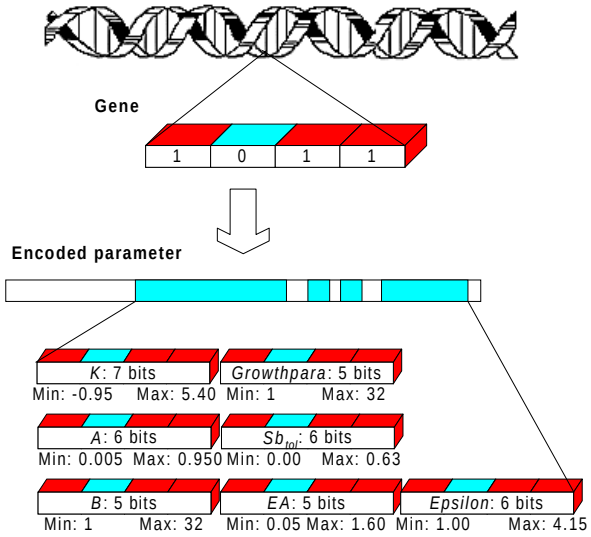

Figure 3. A Genome picture of 7 encoded parameters

5.1 FITNESS EVALUATION

The genome for each individual was decoded into parameter values for CENT. The corresponding model was applied to all nine data sets three times; the resulting trees were evaluated using the two criterion described above, and the overall fitness of an individual calculated from the average fitness of the three runs across the nine data sets.

We investigated two different fitness functions, one in which the two measures were simply combined:

Fitness function$_1$ = Hierarchical distortion-Gamma

And a second which contained an additional factor penalising large trees, namely:

Fitness function$_2$ = Hierarchical distortion–Gamma + (Totalnode if > 40)

In both cases minimisation of the fitness function was required.

6 RESULTS

Figures 4 and 5 show the average and best fitness in the population for the two fitness functions:

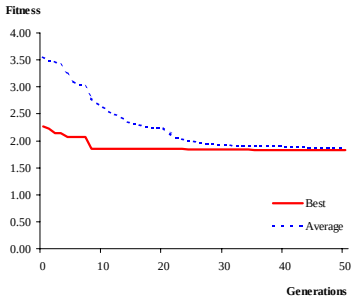

Figure 4. The average and best fitness of the population when using the first fitness function.

It can be seen that in both cases the overall fitness level of the population and the best individual fitness reduced over time. An improvement in performance was obtained by using the parameters found by the GA compared to the set originally thought to be appropriate. For the second fitness function, the final fitness level was significantly higher, due to the propensity of the hierarchical distortion measure to prefer large trees, something heavily penalised by this fitness function. This difference in tree sizes is illustrated in Figures 6 and 7, where, for a representative data set, the total number of nodes and the number of leaf nodes is plotted whenever the best individual changes.

The CENT model is fairly tolerant to its parameter values. Ranges of values can produce reasonable results, hence the GA quickly finds an adequate solution. However a really good solution was harder to find, and the GA needed to run for longer. The relative change in the clustering measures can often be small even though the classification may be visibly better, so that the observed small changes in population fitness can be significant.

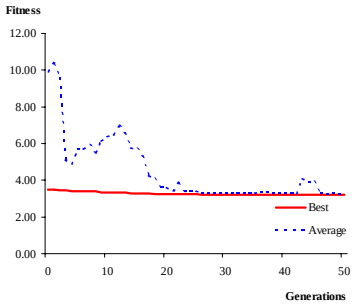

Figure 5. The average and best fitness of the population when using the second fitness function.

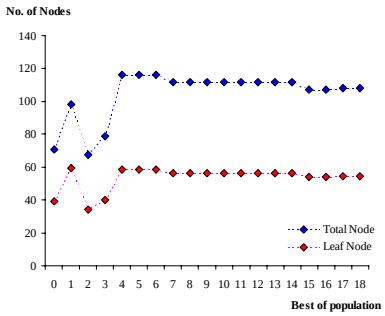

Figure 6. Results in term of number of total nodes, and number of leaf nodes, using the first fitness function and a representative data set.

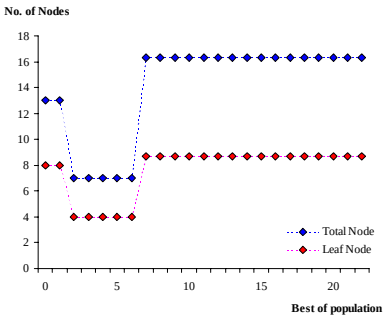

Figure 7. Results in term of number of total nodes, and number of leaf nodes, using the second fitness function and a representative data set.

7 DISCUSSION AND CONCLUSIONS

One of the interesting features of this study is that the fitness function is not applied directly to the genotypes on the population that the GA is evolving but to neural classifiers that have grown in response to environmental stimuli. This provides a direct analogue with the normal genotype to phenotype mapping that occurs in ontogeny.

The results presented here demonstrate that a GA can help in optimizing the internal parameter settings of a neural network. Moreover different fitness functions clearly produced different parameter sets. Previous experimentation had shown that certain parameters had a more critical impact on the tree structures produced, and the GAs produced parameter settings that corresponded with intuition, varying with the aim of the specific fitness function.

Acknowledgments

Part of this research work is financially supported by the Royal Thai Government.

References

[1] Adams, R.G., Butchart, K. and Davey, N. (1999) Hierarchical Classification with a Competitive Evolutionary Neural Tree. *Neural Networks*, Vol. 12, pp 541-551.

[2] Butchart, K. (1996) Hierarchical Clustering Using Dynamic Self Organising Neural Network. *Ph.D. Thesis*. University of Hertfordshire.

[3] Davey,N., Adams, R.G. and George, S.G. (1999) The Architecture and Performance of a Stochastic Competitive Evolutionary Neural Tree Network, *Applied Intelligence*, Vol. 12, No. 1/2, pp.75-93.

[4] Everitt, B.S. (1993) *Cluster Analysis*, Edward Arnold, London.

[5] Gordon, A.D. (1987) A Review of Hierarchical Classification. *Journal of the Royal Statistical Society* ,Vol. 150, pp 119-137.

[6] Grefenstette J.J. (1995) Genesis 5.0, ftp://www.aic.nrl.navy.mil/pub/galist/source-code/ga-source

[7] Hartigan, J.A. (1975) *Clustering Algorithms*, John Wiley & Sons, USA.

[8] Li, T., Tang, Y.Y. and Fang, L.Y. (1995) A Structure-Parameter-Adaptive (SPA) Neural Tree for the Recognition of Large Character Set. *Pattern Recognition*, Vol.28, No. 4, pp 315-329.

[9] Milligan, G.W. and Cooper, M.C. (1985) An Examination of Procedures for Determing the Number of Clusters in a Data Set. *Psychometrika*, Vol. 50, No. 2, pp 159-179.

[10] Mitchell, M. (1998) *An Introduction to Genetic Algorithms*, MIT Press, USA.

[11] Racz, J. and Klotz, T. (1991) Knowledge Representation by Dynamic Competitive Learning Algotithm. *Proceedings SPIE Applications of Artificial Neural Network II* ,pp 778-783.

[12] Song, H. and Lee, S. (1998) A Self-Organising Neural Tree for Large-Set Pattern Classification. *IEEE Transaction on Neural Networks*, Vol. 9, No. 3, pp 369-380.