OPTIMISING A STOCHASTIC DYNAMIC NEURAL TREE

W.Pensuwon[1,2], R.G.Adams[2], and N.Davey[2]

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

[2] Department of Electrical and Electronics Engineering,
Faculty of Engineering, Ubonratchathani University,
34190, Thailand., Tel: +66 045 288376-8

ABSTRACT

This paper describes experiments performed using a Genetic Algorithm (GA) to optimise the parameters of a novel model of a stochastic hierarchical neural clusterer. Two issues of enhancing and optimising the model are discussed. Two fitness functions were created from two selected clustering measures, and a population of genotypes, specifying parameters of the model were evolved. Using the idea of optimising the model by a GA has been proven to be useful. This process mirrors genomic evolution and ontogeny.

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 [1,2].

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 [10].

A more recent DNTN is the Stochastic Competitive Evolutionary Neural Tree (SCENT), [8,9]. This model is comparatively robust, with respect to its parameter settings when compared with other DNTNs [8]. 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 tool. A given genome (a set of parameters), together with a data set allowed SCENT to produce a classificatory tree. Twelve data sets were used. The fitness of these trees was calculated using two clustering measures.

2. STOCHASTIC COMPETITIVE EVOLUTIONARY NEURAL TREE

In SCENT, 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. 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.

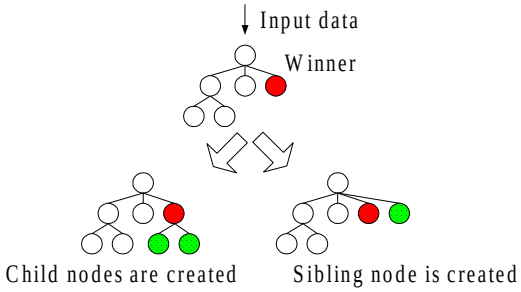
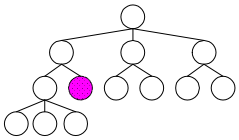
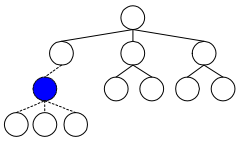


Figure 1. Process of growing a tree

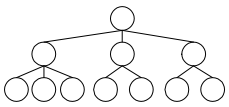
When a node is allowed to grow, if it represents a dense cluster, then its inner counter will be greater than its outer counter and it creates two children. Otherwise, it produces a sibling node. The process of growing is illustrated in figure 1. To improve the tree two pruning algorithms, short and 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. 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.



(a) Node to be pruned.



(b) Singleton is removed, the tree is reconstructed.



(c) Final tree after pruning process.

Figure 2. Pruning process of an inactive node from the tree. The final tree is restructured so that a singleton is removed.

The addition of randomness to a hierarchical neural tree clusterer have some benefit in helping the model avoid local minima in its implicit cost function. This approach is well known in the field of optimisation [9].

There are two different ways in which stochasticity can be added to the model. Firstly the deterministic decisions relating to growth and pruning can be made probabilistic, we call this Decision Based Stochasticity.

Secondly the attributes inherited by nodes when they are created can be calculated with a stochastic element, we call this Generative Stochasticity. To both of these approaches a simulated annealing process can be added to mediate the amount of non-determinism in a controlled way, so that a decreasing temperature allows for less randomness later in the life of the network.

3. PARAMETER SETTINGS

The parameters that are the object of this experiments 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. As *K* is increased, it makes node growth easier, and thereby bigger trees will result.

$$\frac{Node's\ Activity}{Parent's\ Activity} > \frac{1}{Parent's\ NumOfChild + K}$$

The α and β parameters determine the new tolerance of any children produced.

$$NewChildTol = ParentTol \times \left(\alpha + \beta \times \left(\frac{outer}{inner} \right) \right)$$

An increase in α, and/or an increase in β tend to make the trees deeper. When a node produces a sibling these two nodes have larger tolerance values, mediated by *SiblingToleranceSetting*. Increasing this parameter limits sibling growth.

$$NewTolerance = OldTolerance \times (1 + Sib_{Tol})$$

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

$$\frac{(TotalSiblingSumOfError) \times (No.OfSibling)}{Parent's\ PreviousError} > Erroracceptance$$

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 [3].

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 [6].

There are two types of clustering measures, ones that grade the flat clustering performance of the leaf nodes and ones that grade the hierarchical structure.

An initial investigation, concentrated on 10 clustering criterion selected from a comparative evaluation of Milligan and Cooper [6] and another 2 hierarchical methods [4]. From this study, 2 measures were chosen: the gamma measure, which measures the flat partitioning performance and the hierarchical correlation that assesses a hierarchical clustering against the data.

5. OPTIMISING PARAMETERS BY GENETIC ALGORITHMS

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 [7].

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 [5], for 100 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 [7].

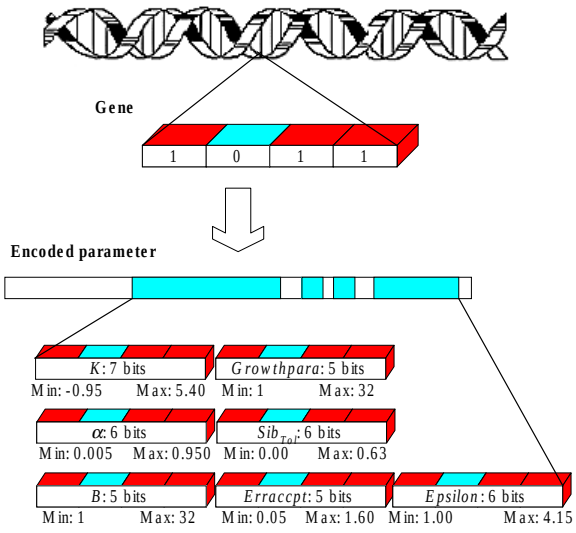


Figure 3. A Genome picture of 7 encoded parameters

5.1. Fitness Evaluation

The fitness function in the GA was constructed in terms of the two best criteria described in the previous section. The gamma measure and the hierarchical measure both need to be maximised. Two fitness functions were used, one just used the 2 cluster measures and was otherwise unconstrained and the other constrained the resulting tree to have fewer than 80 nodes. So the fitness functions are:

Fitness Function1 = Hierarchical Correlation + Gamma

Fitness Function2 = Hierarchical Correlation + Gamma - (Total nodes if > 80)

The genome for each individual was decoded into parameter values for SCENT. The corresponding model was applied to all twelve 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 twelve data sets.

6. RESULTS

Results in this section are divided into 2 sets of results. The first set of results illustrates the convergence of the fitness functions to a final value and gives the sets of parameters found by the best individual over a 100 generation run. These results are shown in Figures 4 and 5. The second set of results is to show the effects on each parameter during the run and are depicted in Figures 6 and 7.

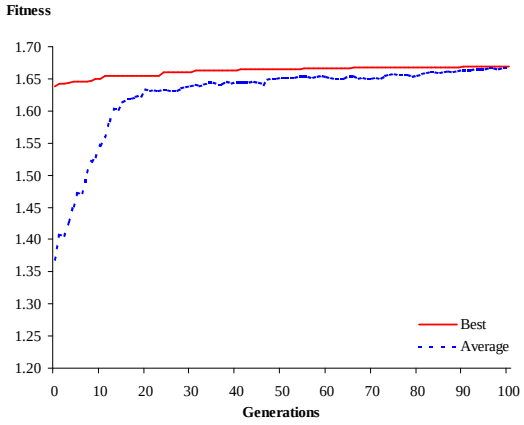


Figure 4. The average and best fitness of the population when using the first fitness function.

Figure 4 depicts clearly that for the first 20 generations there is a big difference in value of the population average fitness and the best individual. Nonetheless, the difference

was reduced as the number of generations increased. Toward the end of the run, the average fitness was increased almost to the level of the best. An analysis of the results for the first fitness function showed that the best individual settled to a set of parameters that always produced network trees with less than 80 nodes.

This is what motivated the creation of the second fitness function. It was designed to see if a reduced search space would produce better results. Since the final result always had less than 80 nodes could a better performance be produced by restricting the whole search space.

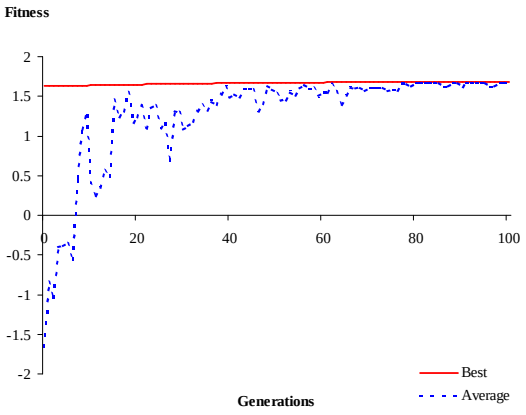


Figure 5. The average and best fitness of the population when using the second fitness function.

Interestingly results show in Figure 5 that the average fitness started with a negative value as it penalised large trees which were more than 80 total nodes in size. However, the average fitness was positive after 10 generations. The average fitness value in Figure 5 was more erratic than that in Figure 4, due to the large penalty associated with large trees.

Nevertheless, this average fitness value also approached the value of the best individual over 100 generations as had been the case with the first fitness function. It can be seen that in both cases the overall fitness level of the population and the best individual fitness increase over time.

During the 100 generations, the best individual in the population changed 19 times. Information regarding each best of population using the first fitness function was recorded and these 19 points represent the x-axis on a graph in Figure 6. In the experimental run using the second fitness function, the best individual only changed 12 times. Again information from each of these individual is plotted in Figure 7 as the change occurred.

Figure 6 and 7 depict the number of total nodes and leaf nodes and the depth of tree when the best of population changed. The 22 clusters data sets was selected to illustrate the changes that occurred to the parameters as each best of population was found. Similar results were obtained for most of the data sets.

From Figure 6, it can be seen that the number of total nodes and leaf nodes and depth of the tree had settled by about the 7th and 8th best of population. In Figure 7, the basic tree shape in terms of number of nodes, number of leaf nodes and tree depth had basically settled by the 5th best of population.

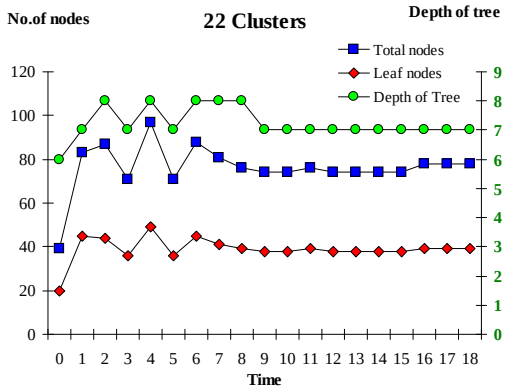


Figure 6. Results in term of number of total nodes and leaf nodes and depth of tree, using the first fitness function and a representative data set.

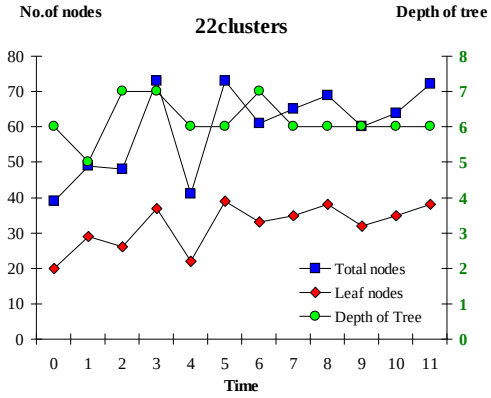


Figure 7. Results in term of number of total nodes and leaf nodes and depth of tree,using the second fitness function and a representative data set.

7. DISCUSSION AND CONCLUSION

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 optimise the network performance by using an indirect coding between the genotype (a set of parameters) and phenotype (the grown tree).

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.

8. REFERENCES

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

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

[3] B.S. Everitt, Cluster Analysis, Edward Arnold, London, 1993.

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

[5] J.J.Grefenstette, Genesis5.0, ftp://www.aic.nrl.navy.mil/ pub/galist/source-code/ga-source, 1995.

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

[7] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, USA, 1998.

[8] W. Pensuwon, R.G. Adams, and N.Davey, Hierarchical Dynamic Neural Networks, Ph.D. Thesis. University of Hertfordshire, 2001..

[9] W. Pensuwon, R.G. Adams, and N.Davey, The Analysis of the Addition of Stochasticity to a Neural Tree Classifier, Journal of Applied Soft Computing, Vol 1 (3), pp 189-200, 2001.

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