An Improved Neural Tree Classifier with Adaptive Pruning

W. Pensuwon
Faculty of Engineering,
Ubonratchathani University,
Thailand
Wanida.Pe@ubu.ac.th

R.G. Adams
Dept. of Computer Science,
Faculty of Eng & Info. Science,
University of Hertfordshire, U.K.
R.G.Adams@herts.ac.uk

N. Davey
Dept. of Computer Science,
Faculty of Eng & Info. Science,
University of Hertfordshire, U.K.
N.Davey@herts.ac.uk

Abstract
This paper describes an improved Stochastic Competitive Evolutionary Neural Tree (SCENT). Adaptive Pruning algorithm is introduced. The idea is to construct the optimal neural tree structure with a two-phase pruning algorithm, which penalise the incompetent clusterer. The comparative results suggest that the method is fairly efficient in terms of simple structure, fast learning speed and relatively high clustering performances.

Keywords
Stochasticity, Dynamic Neural Tree Network, Hierarchical Clustering, Pruning.

INTRODUCTION

The original motivation and objective for using the adaptive pruning, was mainly trying to optimise the clustering performance by penalise the insufficient clusterer. The improved model is improved the idea of the Stochastic Competitive Evolutionary Neural Tree (SCENT) which provides hierarchical classification of unlabelled data. The network produces stable classificatory structures by repeatedly restructuring the insufficient branch of the classification tree based on internal relation between members.

Stochastic Competitive Evolutionary Neural Tree (SCENT) model [7, 8] attempts to combine the advantages of using a tree structure, and of growing the network. SCENT show promise in their ability to produce stable yet flexible hierarchical structures.

The nature of the hierarchical structure and the quality of the final classification produced by the networks is, however, very strongly influenced by the values given to externally set parameters. The SCENT network does not require externally set parameters, it is therefore able to explore data sets without external influence.

STOCHASTIC COMPETITIVE EVOLUTIONARY NEURAL TREE (SCENT)

Top-Level Algorithm

In SCENT [7, 8], the evolution of the neural tree starts the original root node produces a child group which has 2 randomly positioned children. They have tolerance (the radius of its classificatory hypersphere) set to standard deviation of input vectors and its position is set to the mean of input vectors.

There are two counters, calls inner and outer, which count the number time of classified input vector from distance of within or outside tolerance respectively. These counters defined neural tree should grow children or sibling once. It is decisioned that growth is to be allowed. When a node is allowed to grow, 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 growth is illustrated in Figure 1.

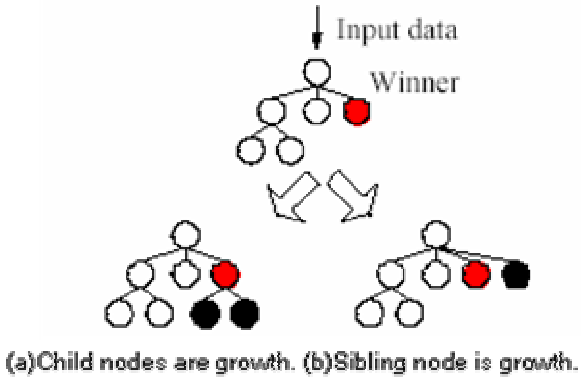


Figure 1. The wining leaf node can grow, (a) children nodes are growth, called downgrowth and (b) sidegrowth in which sibling node is growth.

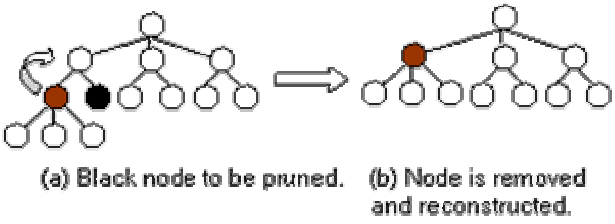


Figure 2. The SCENT models, nodes that are ineffective can be pruned (on the left) when it is also removed (on the right).

This algorithm is called Top-Level Algorithm which a recursive search through the tree is made for a winning branch of the tree. Each node which win of branch is moved top-down towards from input data using the standard competitive neural network update rule. The weight vectors of leaf nodes provide bottom level code-vectors the higher level nodes give representations of the centroids of their corresponding clusters.

Pruning algorithms are divided that two section, short term and long term for efficiently delete. The short-term pruning, nodes are immediately deleted when they are

early life which has error more than set to error acceptance. The long-term pruning delete node when has activity less than a threshold value, illustrated in Figure 2.

Stochasticity

Interest in stochastic method, from the fact that in real world problems, the cost function seldom succeed in defining precisely the real optimality of the solution. The optimality condition is far too complex to be understood by any particular formula, in such cases it is desirable to have methods, which can suggest a set of goods demonstrate such behaviour.

Due to the randomise involved, stochastic method the can handle the cost function and can be considered as promising candidates for solving real world problems.

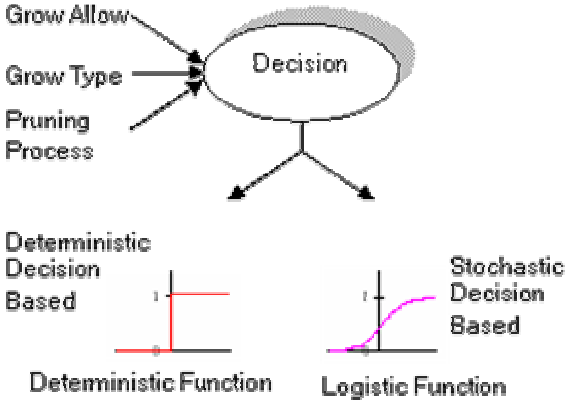


Figure 3. Decision Based Stochasticity. The probability of accepting a decision produced in the left is crisp while the probability of accepting a decision in the right is fuzzy.

One of the main goals of neural network researches has been to build the system the mimic the ability of humans to solve the real world problem. Curiously, tradition approaches in solving have been based on the two valued logic. This is a form of hard computing which based on precision and certainty. It is in sharp contrast to real human reasoning or what is known as common-sense which is based on approximate, rather than precise computing methods.

Adding Stochasticity to a deterministic version (CENT) [1, 2] could have some benefit in helping the model avoid local minima in its implicit cost function. There are two different ways in which stochasticity can be added to the model [8].

Firstly the deterministic decisions relating to growth and pruning can be made probabilistic, it is called Decision Based Stochasticity. The Decision Base Stochasticty has 3 procedures which decisions growth allow, grow type (downgrowth or sidegrowth) and short-term pruning.

Secondly the attributes inherited by nodes when they are created can be calculated with a stochastic element, it is called Generative Stochasticity by changing tolerance values from deterministic function that new tolerance is same every node are created , to stochastic values is randomness effect to new node more flexibility.

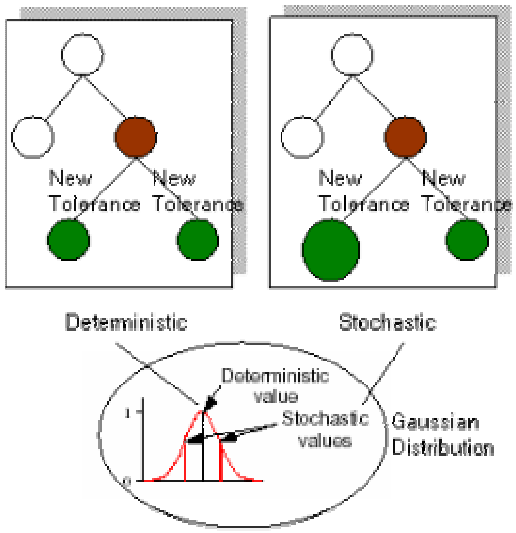


Figure 4. Generative Stochasticity. In deterministic version (CENT) [1, 2] both children have the same value of tolerance (in left crisp). In SCENT a Gaussian is superimposed on the deterministic value to generate 2 different child tolerances (in right crisp).

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.

The Decision will be deterministic which integration of stochastic decision based and simulated annealing. Deterministic version performed well on artificial training set and adding stochasticity still allowed high quality trees to be produced.

AN IMPROVED STOCHASTIC COMPETITIVE EVOLUTIONARY NEURAL TREE (SCENT)

All dynamic networks have to overcome the problem of creating too many nodes and over classifying the data set. These can potentially be a node produced to classify each input vector. In order to prevent over classification, the technique of adaptive pruning in order to create optimal classification nodes is introduced to the original SCENT model.

The improved SCENT has been reengineered in order to improve the clustering performances. There are a few changes in an updated SCENT model in which the pruning process is repeatedly applied after the final process of the classification has been done.

Every branches of classification of the final tree is evaluated by a flat clustering measure. The inadequate branch is deleted and the process of classification will be repeated once to produce a final optimal classification tree.

Algorithm Description

There are 4 major phases in the program: initialisation, main loop, tidy up loop and adaptive pruning loop. In the initialisation phase, the necessary parameters are set, data files are read and a root node is created. In the second phase, the main loop, the tree is allowed to grow and is systematically pruned. The third phase consists of tidying up the final position of nodes in a tree. The final phase, the worse branch of the root's child is deleted and the remaining tree is restructured by repeating clustering process in order to improve its performance. Each phase is described below using pseudocode.

Initialisation

1. Set maximum epoch and learning rate
2. Read parameter file
3. Set *tolerance* of a root node using standard deviation of input vectors
4. Create *RootNode* and 2 initial children nodes and set their *tolerance*

Main Loop

- ***While*** epoch < maximumepoch ***Do***
  - Shuffle inputvectors
  - ***For*** each input vector ***Do***
    - Set *CurrentNode* ← *RootNode*
    - Increment time
    - Update weights, *activity*, and error of *CurrentNode*
    - ***While*** *CurrentNode* ≠ *NULL* ***Do***
      - Find *WinnerNode* from children of *CurrentNode*
      - Update *inner* and *outer* counter of *WinnerNode*
      - Update weights, *activity* and error of *WinnerNode*
      - ***If*** *WinnerNode* is allowed to grow ***Then***
        - ***If*** *outer* > *inner* ***Then***
          - Create **Side-growth** (*WinnerNode*)
          - Initialise *Winner-Node*
        - ***Else***
          - Create **Down-growth** (*WinnerNode*)
          - Initialise *Winner-Node*
      - ***If*** child of *WinnerNode* is not NULL ***Then***
        - Increment level of a tree
        - Set *Currentnode* ← *Winnernode*
      - ***Else***
        - Set *Currentnode* ← *NULL*
      - ***EndWhile***
      - Calculate Sum of Square Error of a tree
      - Update *activity* and error of a tree
  - ***EndFor***
  - Perform **Short-term Pruning**(*RootNode*)
  - Perform **Long-term Pruning**(*RootNode*)

Tidy up loop

- ***While*** epoch < tidyup epoch ***Do***
  - Shuffle inputvectors
  - ***For*** each input vector ***Do***
    - Set *CurrentNode* ← *RootNode*
    - Increment time
    - Update weights, *activity*, and error of *CurrentNode*
    - ***While*** *CurrentNode* ≠ *NULL* ***Do***
      - Find *WinnerNode* from children of *CurrentNode*
      - Update *inner* and *outer* counter of *WinnerNode*
      - Update weights, *activity* and error of *WinnerNode*
      - ***If*** *WinnerNode*'s child is not NULL ***Then***
        - Increment level of tree
        - Set *Currentnode* ← *Winnernode*

- ***Else***

- Set

Currentnode ← *NULL*

- ***EndWhile***

- Calculate Sum square Error of a tree

- Update *activity* and error of a tree

- ***EndFor***

- Perform **Pruning TidyUp**(*RootNode*)

Adaptive Pruning loop

- ***While*** epoch < adaptivepruning epoch ***Do***

- Root's child: Delete the worse branch performance (minimum Gamma Measure)

- Repeat Step Main Loop

- Repeat Tidy up Loop

- Restructure Tree

Clustering 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 [4]. 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 [3, 4, 5]. There are two types of clustering measure: ones that variety the flat clustering performance of the leaf nodes and ones that variety the hierarchical structure.

The Gamma method [3] , which measures the flat partitioning performance, and the Hierarchical Correlation method [5], that assesses the hierarchical structure in a neural tree network. The methods are as follows:

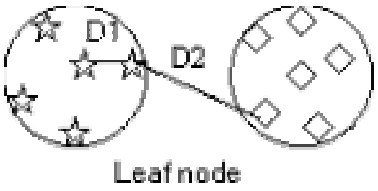


Figure 5. The distance within and between cluster used in calculating Gamma. If D1 is less than D2 then s(+) is incremented and s(-) represents the opposite relationship to s(+).

Gamma: s(+) is number of times when two points not clustered together are further apart than two points which are in the same cluster and s(-) is the number of times when two point not clustered together are closer than two points which are in the cluster. It is compute as,

$$\gamma = \frac{s(+) - s(-)}{s(+) + s(-)}$$

The maximum value indicated the optimal solution. This technique has an upper bound of +1 and a lower bound of -1.

EXPERIMENT

Data Sets

The networks considered in this paper were run over nine different types of data, ranging from a small data set to a large data set. There are two 2-dimensional and two higher dimensional artificial data sets are used. These data sets have cluster structure 4 to 27 classes.

There are nine real world data sets, one example is the well known IRIS data set which consists of 150 instances of 4 attributes describing sepal length, sepal width, petal length and petal width of the iris flower.

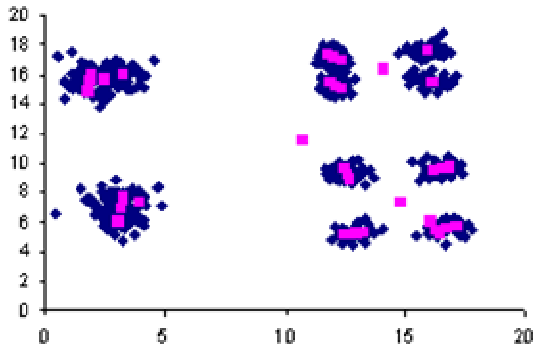
RESULTS

In order to know how well the improved SCENT performs over SCENT, the comparative results of improved SCENT, SCENT and CENT (deterministic model) are presented. The corresponding models were applied to all nine data sets three times; the resulting trees were evaluated using the cluster measure.

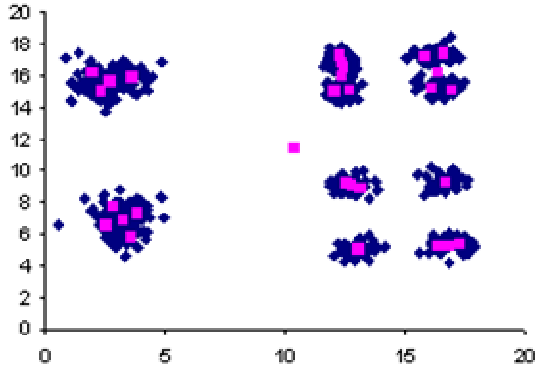
Table 1. Average gamma measures of the classifications produced by the 3 neural network models tested over 9 data sets. The value closed to 1 representing the best performance.

Neural Classifiers	Average	Standard Deviation
CENT	0.827	0.131
SCENT	0.841	0.171
Improved SCENT	0.850	0.147

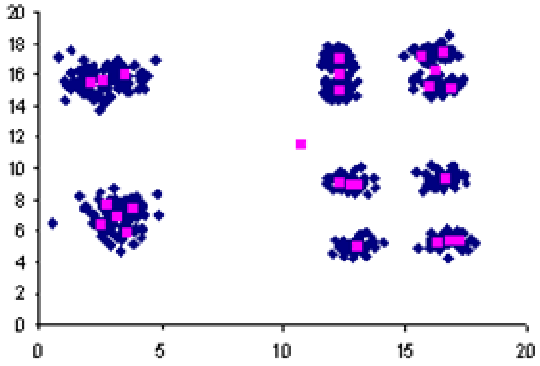
Table 1 represents the average and standard deviation of the gamma measure. The performance of the improved SCENT was clearly better than other two networks. The improved SCENT model produced excellent results in terms of flat and hierarchical clustering measures when viewed as a straightforward cluster, illustrated in Figure 6, giving comparable performance to the SCENT and CENT. It also produced useful, compact and repeatable tree structure so represent any hierarchical information in the data.



(a) CENT



(b) SCENT



(c) Improved SCENT

Figure 6. Leaf node positions of 3 models tested over a representative data set.

CONCLUSIONS

Using neural networks to perform data exploration is difficult; most models require the preimposition of a maximum number of clusters, and will normally classify the data to utilise all classificatory units. There is a question of when to stop training and so when to stop pruning.

Therefore, introducing adaptive pruning by penalise the insufficient classification nodes could balance the scale factor of the misclassification and it can be controlled dynamically.

Using Adaptive Pruning is an interesting and promising approach as a way of taking advantage of the learning advantages of larger systems while avoiding their over fitting problems.

The performance of the propose network (improved SCENT) over a wide range of data sets has been presented and it has been shown to provide a good interpretation of the real world data set, and to produce better classifications compared to the SCENT and the CENT over the nine data sets. The network has shown also that it can handle large data sets.

The improved SCENT model has produced a consistently good performance over all of the data set presented, has maintained the quality of performance and has improved reliability. Further work is being carried out to determine the best combination of restructuring mechanisms for measuring both flat and hierarchical clustering on the stochastic model.

REFERENCES

[1] R.G. Adams. K. Butchart. And N. Davey, "Classification with a Competitive Evolutionary Neural Tree," Neural Networks J, 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," Journal of Applied Intelligence, Vol. 12, No. 1/2, pp. 75-93, 2000.

[3] A. D. Gordon, Classification, Chapman & Hall, London, 1999.

[4] J. A. Hartigan, Clustering Algorithms, John Wiley & Sons, USA, 1975.

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

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

[7] W. Pensuwon, R. G. Adams. and N. Davey, "Comparative Performances of Stochastic Competitive Evolutionary Neural Tree (SCENT) with Neural Classifiers," Proceedings of the 8th International Conference on Neural Information Processing - ICONIP-2001., pp.121-126, 2001.

[8] 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, No.3, pp 189-200, 2002.

(This page left blank intentionally)