DIVISION OF COMPUTER SCIENCE

Investigation of Self-Organising Dynamic Neural Tree Networks

Kate Butchart

Technical Report No.225

May 1995

Investigation of Self-Organising Dynamic Neural Tree Networks

Kate Butchart

Abstract - Self Organising Dynamic Neural Tree Networks (DNTNs) provide hierarchical clustering that is potentially applicable to large data sets. Two DNTN models have been produced by Racz and Klotz and Li et al. Four DNTN variants have been developed and analysed in order to establish which of the four potential cluster expansion methods is the most robust to parameter alterations in the production of representative tree structures.

1. 0 Introduction.

Competitive Learning Neural Networks (CLNNs) have been used successfully for many clustering tasks producing a classification scheme for initially unclassified data [1,2]. CLNNs provide a single level classification where there is no structure or hierarchy amongst the classes produced. Applications such as those involving biological taxonomy require the samples to be classified into hierarchies where the samples in the same group share similar features[3]. Dynamic Neural Tree Networks (DNTNs) perform hierarchically structured clustering, making them particularly suitable for use in areas where a hierarchical classification is required.

Standard CLNNs have a fixed number of output nodes and so classify data according to this fixed number of nodes. If the nature of the data is unknown or may alter, the fixed number of classes can be a limiting factor in the potential success of the model. DNTNs are able to dynamically decide upon the number and structure of the classes required to classify the input data, thus avoiding the problems of deciding upon the number of classes a priori.

Section 2 of the report looks at clustering techniques and standard CLNNs. Section 3 evaluates the principles of DNTNs. The four networks developed are discussed in Section 4. Section 5 discusses the tests carried out and presents the results of these tests.

2 . 0 Clustering and Competitive Learning Neural Networks

Clustering algorithms can be broadly split into two main groups, those using an iterative optimisation approach and those using hierarchical techniques.

Optimisation clustering algorithms reduce a cost function by iteratively altering the fixed number of prototype weight vectors in response to the inputs. The cost function that is being reduced determines the form of the update rule used. The most common cost function is based on the Mean Squared Error (MSE) measure that is given as

$$E = \sum_{x} \left\| x - y_{i^*} \right\|^2 \qquad (1)$$

where x is input vector

y_{i^*} is the winning node

and

$$\left\| x - y_{i^*} \right\|^2 \leq \left\| x - y_i \right\|^2 \qquad \forall i \; (2)$$

A standard CLNN performs clustering on the input data by reducing the MSE given in (1) The MSE is reduced by updating the winning node for each input using the rule

$$y_i(n) = y_i(n-1) + a[x(n) - y_i(n-1)] \quad (3)$$

where a is the learning rate and is global to the whole network.
The update rule in (3) uses gradient descent to minimise the MSE for the fixed number of nodes provided. The number of nodes may not be suitable for the data being clustered or the network may become trapped in a local minima and so the network may provide a poor solution.

The iterative clustering method that CLNNs perform provides a flat, one level representation of the data. In applications such as biological taxonomy it is necessary to provide a hierarchical clustering of the data.

Statistical methods for hierarchical clustering are split into two main groups, agglomerative methods and divisive methods.

Agglomerative methods start with all the individual input vectors representing their own class. A distance matrix between all classes is calculated and the two "closest" classes are merged. This matrix calculation and merging continues until there is either the required number of classes or just one class. Different measures can be used to determine the distance between two classes[4]. The need to calculate an N by N similarity matrix (N is the current number of classes) makes these methods computationally expensive, and so unfeasible, for large data sets especially if the dimensionality of the data is high.

Divisive methods start with all the inputs in one class and split this class until the required number of classes are produced. These methods are either polythetic where all of the elements of the input vector are used to determine the next split, or monothetic where the data is split on the existence or not of just one element of the input vectors. Monothetic techniques are the most computationally efficient of the two techniques but are not always able to produce a suitable classification. Traditional polythetic techniques are computationally expensive and are not feasible for large data sets.

Hierarchic classifications may be represented by dendrogams - as shown in **Figure 1** - which illustrate the fusions or divisions that have been made at each successive stage. The dendrogram shows the fusions or splits and the order in which they occur, and also shows the similarity or distance between the two classes that are split or merged. It should be possible to see the natural clusters in the data from the dendrogram produced.

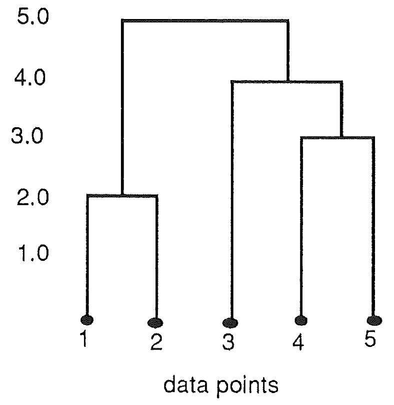

Figure 1: An example dendrogram. The dendrogram shows which of the classes have been combined or split at each stage, and shows the distance or similarity between the two classes that are merged or split. For example when classes 1 and 2 were merged the distance between them was 2 units, whereas the distance between classes 3 and 4 & 5 was 4 units.

3 . 0 Dynamic Neural Tree Networks

DNTNs are single layer feed forward networks where a hierarchical tree structure is superimposed on the output layer nodes. The output nodes are created dynamically and each is fully connected to the input layer. The output nodes are grouped into clusters and the clusters of nodes are organised in a tree structure. The pattern of activity across the output layer is determined in part by the tree structure of the clusters. On presentation of an input to the network only the root cluster is initially active and the nodes within the cluster compete to classify the input. If the winning node has a child cluster this cluster then becomes the current active cluster, this continues until the input is classified by a node with no child cluster.

3.1 Advantages of Dynamic Neural Tree Networks

- Speed of learning - as only subsets of a tree are involved in training at any one time the learning rates are speeded up, especially if the network is large. If N is the number of nodes in the tree, only $P\log_P N$ nodes are explored, where P is the average number of nodes in a cluster.

- Natural hierarchies that exist in the input data set may be replicated in the tree structure of the network. This hierarchical classification may be important in some situations especially those associated with biological data.

- Scalability - statistical methods that perform hierarchical clustering are generally only able to cope with data sets of a limited size. DNTNs may prove to be able to classify much larger data sets.

- Stability - only the group of clusters that were active in the classification of an input are influenced by that input. All of the rest of the network is completely unaffected by the current input. This natural division of the output nodes inherently makes the network stable in response to noisy or changing data.

- Plasticity - as the structure is dynamic it is able to grow or shrink (node deletion is discussed in section 4.1) in response to changes in the input data. This dynamic alteration in the number and structure of the nodes makes the networks more suited to cope with non stationary data than static networks.

DNTNs are able to provide both stability and plasticity, thus they can provide a solution to the stability plasticity dilemma [5]. The networks can grow and shrink in response to changes in the input data without any alteration being required to the classification provided by the rest of the network.

3 .2 Network Structure and Operation

Unlike a dendrogram the final structure of the network is not restricted to a binary tree. An example of a possible tree structure is shown in Figure 2, each node is connected to the input layer, and there are connections between parent nodes and their subtrees. Only one cluster of the total nodes is active at any one time, this cluster is known as the current cluster. The root cluster of the tree is always active initially, and the nodes in that cluster compete to classify the input. The winner of the current cluster is updated using the standard competitive learning rule in (3).

3.3 The production of the Tree structure

DNTNs have associated with them two parameters, known as the tolerance and threshold, which have a strong influence over the structure of the tree created. The tolerance

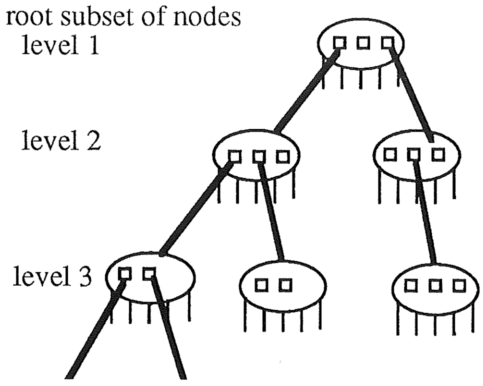

Figure 2: Network Structure

The nodes are grouped into subsets (clusters). Each node in a cluster may be the parent node of another subset of nodes. All nodes are connected to the inputs, but at any one time only one cluster is active

parameter defines the radius of the classifying hypersphere of every node at each level. The size of the tolerance parameter decreases as you travel down the tree away from the root node; so that clusters towards the bottom of the tree provide a finer grained classification than those at the top of the tree.

The threshold parameter determines when a new cluster is created and is described in more detail below.

The tree structure is built dynamically in response to the structure inherent in the data set. The neural tree begins with an empty root cluster. When the first input is presented to the network a node is created in the root cluster to classify the input. This node's weight vector being set equal to that of the first input. As the rest of the inputs are presented to the network one of four situations may occur :

1. The winning node in the current cluster is within the tolerance of the input, and has no child subcluster. This node then classifies that input and moves it weight vector closer to that of the input using the standard rule in (3).

2. The winning node in the current cluster is within a tolerance of the input, and this node does have a child subcluster. The node then learns using (3) and its child cluster becomes the current cluster.

3. The winning node in the current cluster is within the tolerance of the input, has no child subcluster and the threshold value is exceeded. The node then updates its weight vector and creates a new child subcluster.

4. The winning node in the current cluster is not within a tolerance of the input. A new node is then created within the current cluster to classify the input with its weight vector set equal to the input vector.

Cases 3 and 4 cause the tree to grow in response to the data it is classifying.

4. 0 The Four Potential Network Models

Two DNTNs models[6,7] have been produced, A comparative analysis of these models can be found in [8]. The models both use the same Euclidian distance measure to determine when a new node should be produced within a cluster. However the models differ on the heuristic used to determine when a new child cluster should be grown.

4 . 1 Creating a new cluster:

The main aim of the work carried out is to assess the performance of the four possible network variants outlined below in their ability to produce "quality" tree structures. The networks vary only in the choice of stored value for each node and in how these values are compared.

4 . 1. 1 The stored values

The error measure: Each time a node classifies an input the squared error produced by the node in classifying that input is added to the current total error for that node. For each node a quantity known as the total error is maintained.

2. **The activity measure:** Each node has a counter which is incremented each time the node is active.

4 . 1 . 2 Comparing the Values

Absolute comparison: the value stored for each node can be compared to an absolute figure, if it exceeds this figure then a new cluster is created. In order to maintain consistency over time the windowing method of Li et al [7] has been used.

Relative Comparison: The value stored by each node is compared to the value stored by its parent. If the ratio between the values exceeds a preset limit then a new cluster is created.

The Four Network Variants

There are two possible values to store for each node - error and activity , and there are two ways of comparing the measure - to an absolute value or relative to the cluster value. There are therefore four possible different network variants that can be created. These are

1. absolute error network
2. absolute activity network
3. relative error network.
4. relative activity network.

These four networks have been implemented. In order to provide a fair comparison of the different new cluster heuristics the networks were identical in all but this respect. The absolute networks used the window technique defined in Li, and to ensure that the size of the window did not adversely effect the networks performance extra test were run to check this.

4 . 2 Generating new nodes within a cluster

In order for the winning node of the cluster to classify an input the excitation of that winning unit in response to the input must exceed a tolerance α, if the winner's excitation does not exceed this tolerance level then a new unit is generated. In geometric terms if the input is not sufficiently close to any current unit then a new unit is generated to classify it. The tolerance α varies according to the cluster level reducing as you move away from the root. Potential improvements to the method of node generation within clusters are not considered in this study.

5. 0 The Tests

The networks were run over two dimensional data containing structures of hierarchical clusters. Artificially generated two dimensional data was used in order to assist the interpretation of network performance.

The networks were assessed on their ability to:

- Discover the structures present in the data.
- Exhibit robustness to alterations in the tolerance parameter.
- Exhibit robustness to alterations in the threshold parameter
- Exhibit stability in creating an appropriate tree structure.

5. 1 Results

Detailed results are shown in Table 1 and Table 2

All four networks were run over 12 core tests. The first group of tests used a tolerance value of 4.0, which was suitable for the data set being used, with varying threshold values. The absolute networks produced consistent results for all but the highest threshold value where nodes were prevented form expanding thus leading to an unrepresentative distribution of nodes. The relative networks produced good results for mid range threshold values, but low values caused the networks to "run away" - producing over large networks - and high values again caused problems of uneven distribution.

The networks were also tested with tolerance values that were too small (2.0) and tolerance values that were too large (6.0) for the data set. For each tolerance value three threshold values were used - small medium and large. Alterations in the threshold value had the same effect as had been found in the first four tests, i.e. too high a value created uneven expansion and too low a value causes the relative networks to run away.

Tests were also run with very low and very high tolerance values. For the very low value the networks produced only one significant cluster which contained many nodes. The high tolerance value caused the networks to all have a root cluster with just one node.

The absolute networks were run over extra tests to establish what influence the size of the network's window had on the results produced. It was found that provided the window size was not too small (i.e. was not less than a quarter of the size of the data set) it had little effect.

5. 1 Activity versus Error

It was found in these tests that the two measures performed in a very similar fashion. Networks using either measure were able to discover the clusters present in the data set, and performance was in many tests almost identical. There were some differences between the networks and it was found that an error network was more likely to expand a spatially large cluster and an activity network was more likely to expand a spatially compact cluster. However over the data set used these differences were generally confined to the order of cluster discovery as opposed the actual clusters found.

If the network is to be used for Vector Quantization then error has been shown [9,10] to be the most suited to the reduction in MSE. However if the network is to be used for clustering and discovery of the input probability distribution is more important than error reduction then the activity measure may well be preferable as is shown in [9].

5. 2 Absolute or Relative Networks

The absolute networks produced reasonably consistent and reliable results. The threshold parameter determined the depth of the tree, but even with low threshold values the tree did not "run away". These networks showed fair resilience to the threshold parameter settings.

The relative networks were sensitive to the threshold parameter setting. For some values they produced good results. The networks do however have a tendency to "run away" i.e. they continue to produce child clusters when there is no real advantage in doing so. This is a problem inherent in the nature of the relative measure. As the decision on whether to expand or not is based purely on the ratio of the two error (or activity) values and no consideration is given to the fact that the error measure may already be minuscule (i.e. 0.0000001) and that expansion is therefore not suitable. For relative networks to be viable a stopping rule would need to be added to the cluster generation process.
All networks types performed badly when the threshold was set too high as this prevented clusters from being created to represent groups in the data. The tolerance value had a very strong influence over the structure of the network and the results of these tests demonstrates the dependence these networks have on being given suitable parameter values.

References

[1] **Kohonen,T.** (1989).*"Self Organisation and Associative memory"*. Third Edition Springer Verlag.

[2] **Ahalt, S.C. , Krishnamurthy, A. K., Chen, P., and Melton , D.E.** (1990). "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3,no. 3, pp. 277-290.

[3] **Sneath , P.H., and Sokal, R.R.** (1973).*"Numerical Taxonomy, the principles and Practice of Numerical Classification"*, W.H. Freeman and Company, San Francisco

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

[5] **Carpenter G.A, and Grossberg S. ,(1987a),** "A Massively Parallel Architecture for a Self-organising Neural Pattern recognising machine". *Computer Vision, Graphic and Image Processing*, 37 , 54.

[6] **Racz,J. and Klotz , T .** (1991) "Knowledge Representation by dynamic competitive learning techniques." *SPIE Applications of Artificial Neural Networks II* , vol. 1469.

[7] **Li , T . , Tang , Y. , Suen , S . and Fang , L,** (1992). "A structurally adaptive neural tree for recognition of a large character set." In: *Proc. 11th IAPR*

[8] **Butchart, K., Davey, N., Adams, N.**(1995a) ,"A Comparative Study of two Self Organising and Stuctually Adaptive Dynamic NeuralTree Networks", In: *Proceedings ADT'95* to be published.

[9] **Fritzke, B. (1993),** "Kohonen feature maps and growing cell structures - a performance comparison" in *Advances in Neural Information Processing 5*, L. Giles, S. Hanson & J. Cowan, eds., Morgan Kaufman Publishers, San Mateo, CA.

[10] **Ueda, N. and Nakano, R.,** (1994), "A New Competitive Learning Approach Based on an Equidistortion Principle for Designing Optimal Vector Quantizers", *Neural Networks*, Vol.7, No. 8, pp. 1211-1227.

Table 1 Absolute Network results - poor performances are shown in *italics*

		Network	Type
tolerance	threshold	Absolute Activity	Absolute Error
4.0	10.0	4 top level nodes 3 levels Even representation	4 top level nodes 3 levels Even representation
4.0	15.0	3 top level nodes 3 levels Even representaion	3 top level nodes 3 levesl Even representaion
4.0	5.0	3 or 4 top level nodes 4 levels Even representation	3 or 4 top level nodes 3 levels Even representation
4.0	25.0	*3 top level nodes* *2 levels* *Poor representation*	3 or 4 top level nodes 2 levels Even representation
6.0	10.0	2 top level nodes 4 levels Even representation	2 top level nodes 3 levels Even representation
6.0	15.0	2 top level nodes 4 levels Even representation	2 top level nodes 3 levels Even representation
6.0	20.0	2 top level nodes 4 levels Even representation	*2 top level nodes* *2 levels* *Poor representation*
2.0	10.0	9 top level nodes 3 levels Even representation	9 top level nodes 2 levels Even representation
2.0	15.0	*9 top level nodes* *2 levels* *Poor representation*	9 top level nodes 2 levels Even representation
2.0	20.0	*9 top level nodes* *2 levels* *Poor representation*	*9 top level nodes* *2 levels* *Poor representation*
1.0	15.0	*many top level nodes* *2 levels* *poor 2nd level representation*	*many top level nodes* *3 levels* *poor representation*
10.0	20.0	1 node at top level 3 levels Even representation	1 node at top level 3 levels Even representation

Table 2 Relative Network results - poor performances are shown in *italics*

		Network	Type
tolerance	threshold	Relative Activity Network	Relative Error Network
4.0	0.2	3 top level nodes 3 levels Even Representation	3 or 4 top level nodes 3 levels Even representation
4.0	0.25	4 top level nodes 2 levels Even representation	*3 or 4 top level nodes* *2 to 6 levels* *Poor representation*
4.0	0.1	*3 top level nodes* *6 levels* *Poor representation*	*3 top level nodes* *many levels* *poor representation and cluster proliferation*
4.0	0.15	*3 or 4 top level nodes* *5 levels* *Poor representation*	*3 or 4 top level nodes* *8 levels* *poor representation and cluster proliferation*
4.0	0.35	*3 or 4 top level nodes* *2 levels* *Variable representation*	*3 to level nodes* *2 levels* *poor representation*
6.0	0.15	2 top level nodes 3 levels Even Representation	*2 top level nodes* *8 levels* *poor representation and cluster proliferation*
6.0	0.1	2 top level nodes 4 levels Even Representation	*2 top level nodes* *9 levels* *poor representation and cluster proliferation*
6.0	0.25	2 top level nodes 2 levels Even representation	*2 top level nodes* *2 to 9 levels* *Poor representation*
2.0	0.1	*9 top level nodes* *7 levels* *cluster proliferation*	*9 top level nodes* *12 levels* *cluster proliferation*
2.0	0.15	9 top level nodes 5 levels Even representation	*9 top level nodes* *12 levels* *cluster proliferation*
2.0	0.25	9 top level nodes 1 level	9 top level nodes 1 level
1.0	0.15	many top level nodes 1 level	many top level nodes 1 level
10.0	0.2	1 top level node 2 levels Fair representation	1 top level node 4 levels Fair representation