Hierarchical Growing Neural Gas

K.A.J. Doherty, R.G. Adams, N. Davey
Department of Computer Science, University of Hertfordshire, United Kingdom
E-mail: {K.A.J.Doherty, R.G.Adams, N.Davey}@herts.ac.uk

Abstract

This paper describes TreeGNG, a top-down unsupervised learning method that produces hierarchical classification schemes. TreeGNG is an extension to the Growing Neural Gas algorithm that maintains a time history of the learned topological mapping. TreeGNG is able to correct poor decisions made during the early phases of the construction of the tree, and provides the novel ability to influence the general shape and form of the learned hierarchy.

1 Introduction

The discovery of hierarchical structure through statistical methods is generally referred to as cluster analysis or numerical taxonomy, and these techniques are well established. The statistical generation of a hierarchical clustering can be achieved by either agglomerative or divisive methods. However, the divisive methods suffer from the inability to recover from a poor decision in the construction of the dendrogram [1].

Unsupervised Competitive Learning is the artificial neural network foil to cluster analysis. A sub-set of the family of Unsupervised Competitive Learning methods are the Growing Self-organising networks. Each node of the network has a position vector in the input space, and nodes are connected by edges to form graphs. The competitive Hebbian rule [2] produces the edges that form a sub-set of the Delaunay triangulation for the nodes [3]. The neighbourhood of a node is defined by the edges incident to the node. The positions of the nodes are altered in response to each input, and the structure of the network is modified by the insertion and deletion of nodes and edges. This dynamic behaviour can result in disjoint graph structures. The procedure used to form the graph and the resultant graph structure are called topology representing networks (TRN) [4].

TRN research has mainly focused on the discovery of concepts without hierarchical structure, but there have been recent attempts to discover and learn the taxonomy of concepts contained in an unlabelled set of data. A family of hierarchical neural clusterers has emerged based on Fritzke's Growing Cell Structures (GCS) [5]. However, the GCS algorithm has some inherent problems, which we will describe later in this document.

In this paper we propose a new unsupervised hierarchical, top-down classifier. Our model uses the Growing Neural Gas (GNG) algorithm [6], removing the reliance on the sometimes less-than-successful partitioning produced by GCS, and provides the ability to alter the general shape and form of the tree structure. The remainder of this paper is organised as follows: In the next two sections, we describe the dynamics and performance of the GCS and GNG networks, and the known hierarchical variants. In section 4, we present our model and the results of our experiments, and in the final section, section 5, we draw our conclusions.

2 Topology Representing Networks

In this section, we very briefly describe the dynamics, and comment on the performance, of the GCS and GNG networks.

2.1 Growing Cell Structures

The GCS algorithm grows a network composed of k-dimensional network construction units, for which k is user-defined (fig. 1). Since k is generally less than that of the arity of the input data, the GCS model performs a dimensionality-reducing mapping from the (possibly high dimensional) input space into a (generally lower) user-defined dimensional output space.

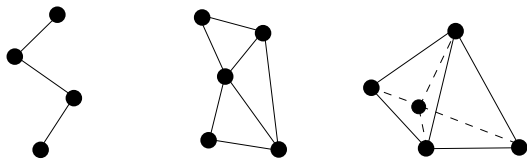

Fig. 1. Examples of GCS network topologies for k-dimensional network construction units [5].

Periodic node deletion occurs based on a measure of node activity and the volume of the input space classified by the node. The calculation of the Voronoi volume required for node deletion is difficult in dimensions greater

than 2, and algorithm implementations have resorted to estimates [7, 5].

2.2 Growing Neural Gas

Growing Neural Gas is similar to GCS but does not impose the strict network-topology preservation rule, and edges are deleted based on an age criterion. The network incrementally learns the Delaunay triangulation corresponding to the topological relationships inherent in the data set, and continues until a user defined stopping criterion is satisfied.

2.3 GCS and GNG Comparison

Fritzke claimed that GCS "automatically finds meaningful partitions of the data", and that GCS was able to estimate the probability density of the input "under a wide range of parameter settings" [5]. In later work, Köhle and Merkl used GCS for document classification. Their results suggested that the GCS automatic cluster boundary generation aided the identification of cluster structure. However, they also noted that the algorithm is very sensitive to parameter selection, and for a wide range of parameters, GCS was "unable to produce semantically meaningful classification results" [8]. The strict topological preservation rule of GCS can result in massive purges in the GCS network, causing much of the accumulated learning to be lost [9]. The results of our own investigations also suggest that GCS clustering results are heavily dependent on the network parameter settings.

In a performance comparison of three incremental networks (including GNG and GCS) and the multilayer perceptron, the networks were benchmarked on four datasets, and scored for classification error, convergence rate and parameter sensitivity [10]. The GNG algorithm returned the superior benchmark score. The algorithm converged rapidly and showed little dependence on the network parameter settings. Again, the results of our own experiments agree with these findings.

3 TreeGCS

Various hierarchical variants to GCS and GNG have been proposed [9, 11]. TreeGCS [7] is an interesting variant. TreeGCS is a top-down, incremental learning hierarchical classifier, that maintains a time history tree of the graph connectivity of a standard GCS algorithm. As a part of the normal GCS dynamics, periodic node deletion takes place, occasionally resulting in graphs splitting into two or more disjoint graphs. Every disjoint GCS graph structure is represented by a leaf node in the tree representation. Every epoch, the tree representation is examined, and if a GCS graph structure represented by a leaf node has split, then a new child for every new graph is inserted into the hierarchy beneath the old leaf node. Similarly, if a disjoint sub-graph is deleted from the network, then the leaf node associated with the sub-graph is removed from the tree, and the hierarchy is updated to remove any inconsistent structure e.g. nodes with a single child are removed.

4 TreeGNG

Based on the results of our own experiments and the work of others, we consider that the relatively poor performance of GCS should preclude its use as the underlying algorithm. We believe that an improved basis for this time mapping approach would be the GNG algorithm. We propose the TreeGNG algorithm (fig. 2), which follows the TreeGCS algorithm but uses GNG as the underlying clustering algorithm, has a user-defined graph generation rate and replaces the epoch count stopping criterion with a more flexible user-defined stopping criterion.

```
Until stopping criterion is satisfied
 For each input
  Run GNG, generate graph structures
  If (Tree generation time)
   If (No. of clusters increased)
    Identify the tree node that now
    points to multiple clusters,
    create new children for this node
    and associate the new children
    with the clusters
   Elseif (No. of clusters decreased)
    Remove the associated tree node
    from the hierarchy and ensure the
    tree structure contains no
    singletons
   End if
  End if
 End for
End program
```

Fig. 2. The TreeGNG Algorithm

Within the GCS algorithm, the rate of node deletion is a user-defined iteration count; within the TreeGCS algorithm, the generation of the tree structure is based on an epoch-by-epoch examination of the graph connectivity. The same approach could be followed for TreeGNG. However, we decided that we could make use of the fact that the shape of the final tree will be dependent on the relationship of these two key periods. If *tree generation* occurs very infrequently in relation to the frequency of *node deletion*, then the tree will be very shallow and possibly have a large branching factor. As tree generation becomes more frequent, the tree will tend to get deeper

with a smaller branching factor, and taken to the limit, if the tree is generated every time a change occurs in the number of graphs, then the tree will be binary. This aspect was not examined by Hodge and Austin, but we believe it provides a useful tool, as the the general shape and form of the tree can be specified.

Fig. 3 illustrates the TreeGNG graph splitting and the resultant tree growth.

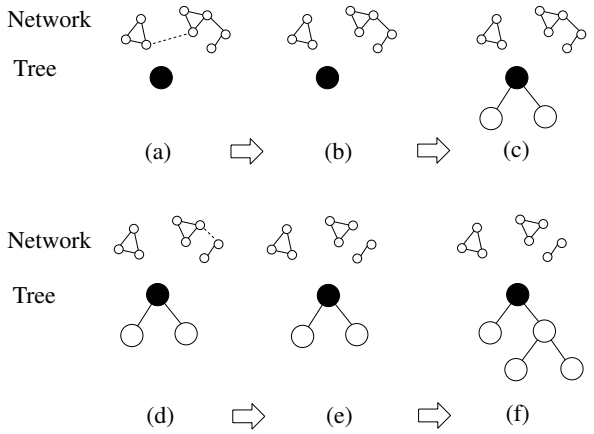

Fig. 3. GNG graph splitting and tree growth. In network (a), following the standard GNG ageing dynamics, the dashed edge is marked for deletion. Edge deletion results in two disjoint networks (b). At tree generation time, the tree is updated with new subordinate nodes (c) to reflect the the splitting of the graph. The edge ageing and deletion process is repeated in graphs (d) and (e) producing further tree growth (f) at the next tree generation time.

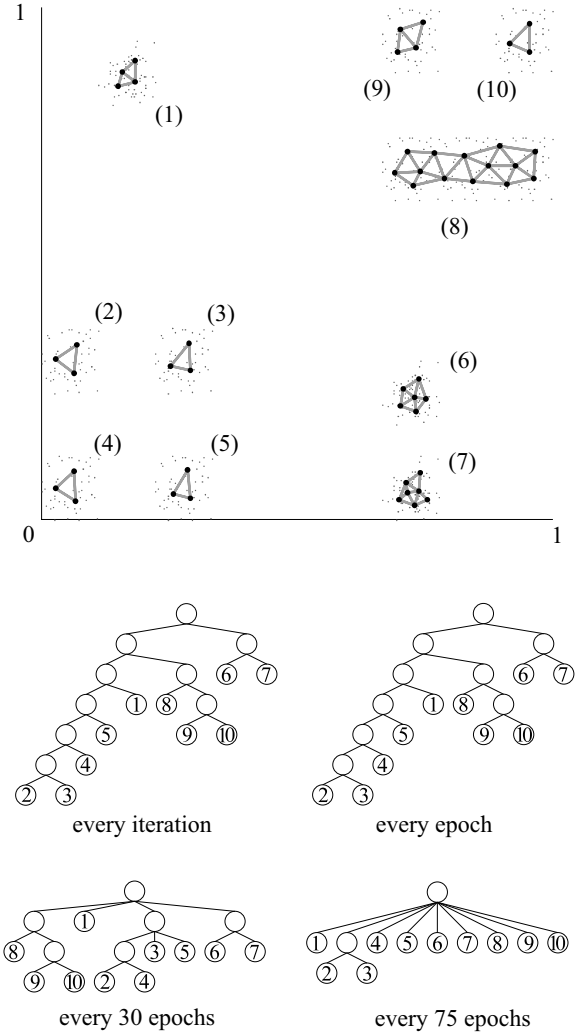

Fig. 4. GNG clustering of the MIX10 data set (upper fig.) with 50 nodes, a GNG edge deletion age of 1 epoch and a node insertion rate of 2 per epoch. The time history trees for a range of tree generation intervals are shown in the lower figures. With the tree generation occurring every 30 epochs, the tree structure was broadly in-line with our expectations of an appropriate hierarchical representation.

4.1 Experimental Results

We tested the utility of our algorithm on a range of data, and report the results for two synthetic data sets. The data sets comprised of 675 and 900 elements in $\mathbb{R}^2$. For both data sets, we ran the GNG algorithm for 5000 epochs with a range of parameters. For all the parameters considered, GNG satisfactorily clustered these data in-line with our expectations, and the induced Delaunay triangulation indicated the appropriate number of discrete clusters.

Figs. 4 and 5 (upper) shows the results of clustering with 50 codebook vectors, for a GNG edge deletion age of 1 epoch and 2 nodes insertions per epoch. It should be noted that the overall clustering time can be reduced by using a smaller edge deletion age, with little (if any) impact on the quality of the final clustering. We recorded the time history of the graph connectivity for tree generation intervals ranging from 1 iteration to 100 epochs, and typical resultant tree structures are shown (figs. 4 and 5 lower). The trees confirm that frequent tree generation results in binary tree representations, whilst an extended period between tree generations results in wider, shallower trees; but of more importance is that between these two extremes, the tree structure is in-line with our expectation of the most appropriate tree structure. In addition, the TreeGNG tree structures do no exhibit the instability noted with TreeGCS where "for many parameter settings" the network repeatedly deletes and reinstates the

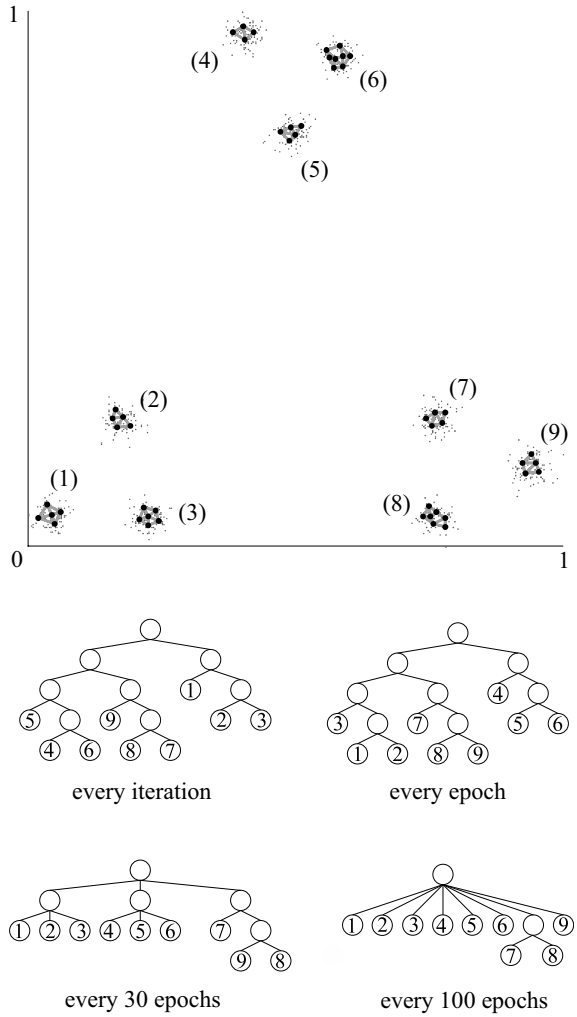

Fig. 5. GNG clustering of the GAUS9 data set (upper fig.) with 50 nodes, a GNG edge deletion age of 1 epoch and a node insertion rate of 2 per epoch. The resultant time history trees for a range of tree generation intervals are shown in the lower figures. With the tree generation occurring every 30 epochs, the tree structure was broadly in-line with our expectations of an appropriate hierarchical representation.

same clusters [7]. The results of tests on other synthetic data sets have indicated that TreeGNG is able to produce trees in-line with our expectations of an approriate structure.

5 Conclusions

We have developed an unsupervised top-down hierarchical classification tool based on Fritzke's GNG algorithm and Hodge and Austin's TreeGCS. Our method dynamically learns and adjusts the tree structure in response to the input data, and produces stable hierarchical representations for a broad range of network parameters. The periodic edge removal and competitive Hebbian learning of the GNG algorithm allows the network to recover from poor decisions in the generation of the hierarchy and thus overcome one of the major problems with TreeGCS and the divisive statistical methods. The algorithm also provides the novel ability to influence the general shape and form of the learned hierarchy.

References

[1] Everitt, B. (1993), Cluster Analysis, Edward Arnold, London, 3rd edition

[2] Hertz, J., Krogh, A., Palmer, R. (1991), Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA

[3] Fritzke, B. (1996), Growing Self-Organizing Networks-Why?, In: Proc. European Symposium on Artificial Neural Networks, pp. 61–72

[4] Martinetz, T. M., Schulten, K. J. (1994), Topology Representing Networks, Neural Networks, 7(3):507–522

[5] Fritzke, B. (1994), Growing Cell Structures - A Self-Organising Network for Unsupervised and Supervised Learning, Neural Networks, 7(9):1441–1460

[6] Fritzke, B. (1995), A Growing Neural Gas Network Learns Topologies, Advances in Neural Information Processing Systems, pp. 625–632

[7] Hodge, V. J., Austin, J. (2001), Hierarchical Growing Cell Structures: TreeGCS, IEEE Trans. Knowledge and Data Engineering, 13(2):207–218

[8] Köhle, M., Merkl, D. (1996), Visualising Similarities in High Dimensional Input Spaces with a Growing and Splitting Neural Network, Lecture Notes in Computer Science, 1112:581–586

[9] Burzevski, V., Mohan, C. K. (1996), Hierarchical Growing Cell Structures, In: Proc. IEEE Int. Conf. Neural Networks, Washington D.C., volume 3, pp. 1658–1663

[10] Heinke, D., Hamker, F. H. (1998), Comparing Neural Networks: A Benchmark on Growing Neural Gas, Growing Cell Structures, and Fuzzy ARTMAP, IEEE Trans. Neural Networks, 9(6):1279–1291

[11] Cao, X., Suganthan, P. (2002), Hierarchical Overlapped Growing Neural Gas Networks with Applications to Video Shot Detection and Motion Characteristics, In: Proc. Int. Joint Conf. Neural Networks, IEEE, Hawaii, USA, volume 2, pp. 1069–1074