World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:10, 2008

Towards Growing Self-Organizing Neural Networks with Fixed Dimensionality

Guojian Cheng, Tianshi Liu, Jiaxin Han, and Zheng Wang

***Abstract*—**The competitive learning is an adaptive process in which the neurons in a neural network gradually become sensitive to different input pattern clusters. The basic idea behind the Kohonen's Self-Organizing Feature Maps (SOFM) is competitive learning. SOFM can generate mappings from high-dimensional signal spaces to lower dimensional topological structures. The main features of this kind of mappings are topology preserving, feature mappings and probability distribution approximation of input patterns. To overcome some limitations of SOFM, e.g., a fixed number of neural units and a topology of fixed dimensionality, Growing Self-Organizing Neural Network (GSONN) can be used. GSONN can change its topological structure during learning. It grows by learning and shrinks by forgetting. To speed up the training and convergence, a new variant of GSONN, twin growing cell structures (TGCS) is presented here. This paper first gives an introduction to competitive learning, SOFM and its variants. Then, we discuss some GSONN with fixed dimensionality, which include growing cell structures, its variants and the author's model: TGCS. It is ended with some testing results comparison and conclusions.

***Keywords*—**Artificial neural networks, Competitive learning, Growing cell structures, Self-organizing feature maps.

I. Introduction

THE learning in ANN is achieved by proper adjustment of the interconnection weights between artificial neurons or by change of network topologies. The ANN learning algorithms can be classified into three main paradigms: supervised learning, unsupervised learning and reinforcement learning. Most of the competitive learning algorithms are used for unsupervised learning in which the input data can be categorized or clustered, i.e., similar inputs are classified as being in the same category and should activate the same output unit which corresponds to a prototype of the category. In this case, the output sample is absent or not needed. Based on correlations of the inputs, clusters are determined by the network itself.

ANN with unsupervised learning has been widely used in clustering tasks, dimensionality reduction, data mining, information extraction, density approximation, data compression, etc. A basic principle of unsupervised learning is based on the competition mechanism, in which the output units compete for activation. In some competitive learning algorithms only one output neuron is activated at any given time, which is realized by means of the so-called Winner-Takes-All (WTA) mode (e.g., LBG and K-means) or Hard Competitive Learning (HCL). Another widely used learning mode is Winner-Takes-More (WTM), i.e., Soft Competitive Learning (SCL). The WTM mode is characterized by adapting in addition to the winner also some other units located at the winner's neighbors. It can be further classified by its topology, WTM without fixed dimensionality and WTM with fixed dimensionality [1].

WTM without fixed dimensionality has no topology of a fixed dimensionality that is imposed on the network. The dimensionality of the network depends on the local dimensionality of the data and may vary within the input space. Competitive Hebbian learning, neural gas and growing neural gas belong to this type. WTM with fixed dimensionality has such an advantage that its network defines a mapping from the n-dimensional input space (with n being arbitrarily large) to the k-dimensional network structure (with k being 2 or 3). This makes it possible to get a low-dimensional representation of the input patterns that may be used for data visualization purposes. The dimensionality k has to be chosen in advance. This kind of models includes Kohonen's Self-Organizing Feature Maps (SOFM) [2] and Fritzke's Growing Cell Structures (GCS) [3]. This paper stresses the GSONN with fixed dimensionality. In the following section 2, the SOFM and its dynamic topology variants are introduced. In sections 3, we present a typical GSONN with fixed dimensionality, GCS and its variants. The author's model, Twin GCS, is presented in section 4. Some testing comparisons with different models are given in section 5. It is ended with conclusion. Notes: in the context, such concepts as neuron, cell, node, unit, element, vertex, etc. have nearly same meaning.

II. Self-Organizing Feature Maps and Variants

The basic idea behind the SOFM is competitive learning. The neurons are presented with the inputs, which calculate their weighted sum and neuron with the largest output is chosen to receive additional training. Training in SOFM does not just affect the one neuron but also its neighbors. Suppose that an input pattern has n features and is represented by a vector $\boldsymbol{x}$ in an n-dimensional pattern space. The network maps the input pattern to an output space. The output space is supposed to be a 1-dimensional or 2-dimensional array of output nodes. The question is how to train a network so that the ordered relationship can be preserved. The cerebral cortex of the human brain could be imaged as a 2-dimensional plane of neurons and spatial mappings are used to model complex inputs. This means that topological relationships in external stimuli are preserved and complex high-dimensional data can

This work is supported by NSFC -- The National Natural Science Foundation of China (40572082).

Authors are with School of Computer Science, Xi'an Shiyou University Shaanxi Province, 710065, China ({gjcheng,tshliu,jxhan,zhwang}@xsyu.edu.cn).

be represented in a lower dimensional space. Kohonen uses 2-dimensional networks where the neurons are arranged on a flat grid in some regular topology, e.g., hexagonal or rectangular. The weights on lateral interconnections from neighbors are given by a function which radius decreases with time, e.g., the Mexican hat function. In other words, the connections from close neighbors should have high (excitatory) weighting and those from distant neighbors should have low (inhibitory) weighting. Therefore, the SOFM is also called a topology-preserving map because it can preserve the neighborhood relations.

Kohonen has also used another learning technique for his self-organizing network [2]. He calls this technique Learning Vector Quantization (LVQ) [2] and uses it to fine-tune the feature map using input vectors of known classification. These vectors are presented to the network and a best match comparison made at each network cell. The weight vector of the winner is modified to reinforce correct classifications or correct misclassifications. Based on LVQ, A. Zell represented a dynamic extension of LVQ, which is called Dynamic LVQ (DLVQ) [4]. Combining the SOFM with a counter-propagation network, J. Goppert and W. Rosenstiel introduced Interpolation SOM (ISOM) [5].

The key advantage of SOFM is the formation of clusters, which helps to reduce the input space into representative features using a self-organization process. But it has some disadvantages. For example, the SOFM uses a fixed network architecture in terms of number and arrangement of neural processing elements which has to be defined prior to training. Obviously, in case of largely unknown input data characteristics, it is very difficulty to predetermine a proper network architecture that can yield satisfying results. Also, the topology of the input space in SOFM has to match the topology of the output space to be represented, i.e., the property of neighborhood preservation depends on the choice of the output space map topology. However, in real world datasets, the proper dimensionality required by the input space is usually not known a priori, yet the output grid size has to be specified prior to learning. To solve this problem, we can use an advanced learning scheme, by which it adapts not only the weight vectors of the neurons, but also the topology of the output space itself. Some examples of such learning algorithms include Self-Organizing Surfaces (SOS) [6], Evolve Self-Organizing Maps (ESOM) [7], Incremental Grid Growing (IGG) [8], Growing Hierarchical Self-Organizing Map (GHSOM) [9].

III. Growing Cell Structures and Variants

B. Fritzke has identified some limitations of SOFM, such as: (a) The SOFM network has a fixed size and topology which must be predetermined; (b) The parameters in the SOFM are not constant over time but follow a decay schedule; (c) Some cells of the SOFM network are at locations where the probability density $p(\boldsymbol{x})$ of the input vector $\boldsymbol{x}$ is zero; (d) Some cells that are direct topological neighbors in the SOFM network have rather distant positions in $p(\boldsymbol{x})$. These limitations result in poor performance for pattern clustering in using SOFM for some applications. To overcome these limitations, B. Fritzke proposed a growing model for SOFM, i.e., Growing Cell Structures (GCS) [3].

A. Principle of GCS

The basic building blocks of the generated topology in GCS are hyper-tetrahedrons of a certain dimensionality k chosen in advance. In contrast to SOFM, neither the number of cells nor the exact topology has to be predefined in GCS. Instead, a growth process successively inserts cells and connections. All parameters in the model are constant over time. This makes it possible to continue the growth process until a specific network size is reached or until an application-dependent performance criterion is fulfilled. The input data directly guide the insertion of new cells. Generally, this leads to network structures reflecting the given input distribution better than a predefined topology could. The purpose of the GCS is the generation of a topology-preserving mapping from the input space $\boldsymbol{R}^n$ onto a topological structure $\boldsymbol{A}$ of equal or lower dimensionality k. Topology preservation has such meanings as: (a) Input vectors that are close in $\boldsymbol{R}^n$ should be mapped onto neighboring nodes in $\boldsymbol{A}$; (b) Neighboring nodes in $\boldsymbol{A}$ should have similar input vectors from $\boldsymbol{R}^n$ mapped onto them.

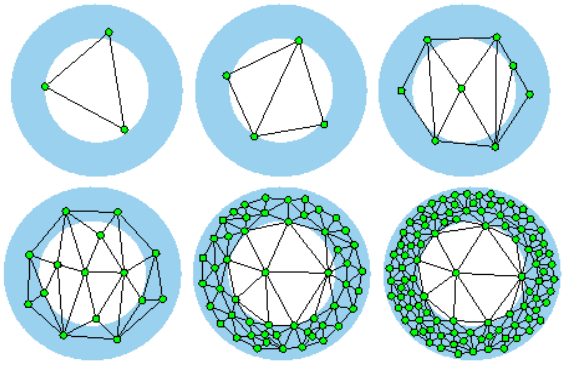

Fig. 1 GCS simulation sequence from initial state to final state for a ring-shaped uniform probability distribution

The GCS generates a k-dimensional topological structure that can be seen as a projection onto a nonlinear, discretely sampled subspace. The GCS starts with a triangle of three cells at random positions in $p(\boldsymbol{x})$, cells are positioned over the area of non-zero probability density. The insertion of new cells should maintain the triangular connectivity structure. The resulting structure is then redistributed through an adaptive process which is analogous to SOFM, i.e., input vectors are generated according to $p(\boldsymbol{x})$, then the best matching unit (V_{bmu} or *winner*) and its direct topological neighbors (V_{dtn}) move closer to this vector. The degree by which the winner and its *dtn* (*direct topological neighbor*) cells are adapted by constant values, η_{win} and η_{dtn} respectively, here η_{win} is more greater than η_{dtn}. The result is a network structure which has reached an equilibrium state where the cell distribution roughly approximates $p(\boldsymbol{x})$. Fig. 1 shows some stages of a simulation for a simple ring-shaped data distribution [1].

B. GCS Variants

Dynamic Cell Structure (DCS) was introduced by J. Bruske [10]. It belongs to the family of Topology Representing Networks (TRN) [11]. DCS employs a modified Kohonen learning rule in conjunction with Competitive Hebbian Learning. The Kohonen type learning rule serves to adjust the synaptic weight vectors while Hebbian learning establishes a dynamic lateral connection structure between the cells reflecting the topology of the feature manifold. The DCS idea applied to the GCS algorithm could lead to an efficient and elegant algorithm.

Based upon hierarchical clustering and GCS, a Tree-based GCS (TreeGCS) was proposed by V. Hodge [12]. TreeGCS is an unsupervised, growing, self-organizing hierarchy of nodes able to form discrete clusters. High dimensional inputs are mapped onto a 2-dimensional hierarchy reflecting the topological ordering of the input space. The TreeGCS algorithm can improve an inconsistency in the GCS algorithm, where the network topology is susceptible to the ordering of the input vectors.

A probabilistic version of the GCS algorithm, Probabilistic GCS (PGCS), was introduced by N. Vlassis [13] and was also applied to a robot configuration. The original GCS is actually an adaptive-means clustering algorithm in which new clusters are added dynamically to produce a Voronoi tessellation of the input space. Under the condition that samples are distributed in each cluster according to a multivariate normal probability density function, the non-parametric model of the GCS was extended into PGCS. By recursively estimating the means and the variances of the clusters, and by introducing a new criterion for the insertion and deletion of a cluster, the PGCS has shown to be more powerful than the original GCS.

Some new variants and application fields of GCS have been presented and it is shown that GCS is an effective and promising GSONN. Those examples include: the evolving tree model by J. Pakkanen [14], adaptive self-organizing maps by Y. Yang [15], adaptive topological tree structure by R. Freeman [16].

IV. Twin Growing Cell Structures (TGCS)

In each insertion step of the GCS only one new cell is inserted in the middle of the edge connecting *MRV* (*Maximum Resource Vertex,* some quantitative errors can be used as a resource for cell growing) and the most distant node in the direct topological neighborhood of *MRV*. But in the TGCS, two new cells are inserted at the same time. The first new cell is inserted in the same way as in GCS. Beside that the second new cell is inserted between the edge connecting the second *MRV* and its most distant direct topological neighborhood. The goal of TGCS is to speed up the convergence of the learning process. Another characteristic of the TGCS is that the insertion point of the new cells is variable. The exact location of their centers of receptive field is calculated according to the ratio of the resource values. This gives an estimate of the resource values as if the new unit has been in the network right from the start.

A. TGCS Topology

The cell growing mechanism of TGCS is nearly the same as in GCS. The only difference is that in TGCS two new cells are inserted simultaneously instead of only one new cell as in GCS. In other words, in each insertion step of GCS only one new cell is inserted in the middle of the edge connecting the *MRV* and the most distant node in the direct topological neighborhood of *MRV*. But in TGCS two new cells are inserted at the same time. The first new cell is inserted in the same way as in GCS. Besides that, another new cell is inserted between the edge connecting the second *MRV* and the most distant node in its direct topological neighborhood (see Fig. 2). When the second *MRV* is in the direct topological neighborhood of the first *MRV*, only one new cell is inserted between first *MRV* and second *MRV.*

Each vertex V_c in the current network $\boldsymbol{A}$ maintains an error counter value $\boldsymbol{E}_c$ over each organizational process. Whenever an input vector is mapped onto a vertex, the square of the distance between the input vector $\boldsymbol{x}$ and the vertex's weight vector $\boldsymbol{w}_{win}$ of the winner cell is added to the error value. Large cumulative error values occur at vertices that have too many input vectors mapped onto them. Their weight vectors fail to adequately represent all of the input vectors in that area. Therefore, new nodes are added to the areas with high cumulative error.

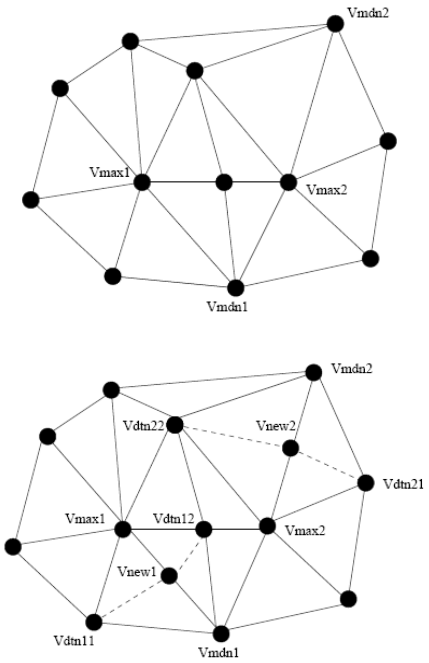

Fig. 2 TGCS network topology. Upper figure: before the insertion of new cells. Lower figure: after the insertion of new cells

Fig. 2 shows the structure of a 2-dimensional TGCS. V_{max1} is the first *MRV*, and V_{max2} is the second. V_{max1} has three direct topological neighbors: V_{dtn11}, V_{dtn12} and V_{mdn1}, V_{max2} also has three direct topological neighbors: V_{dtn21}, V_{dtn22} and V_{mdn2}.

V_{mdn1} and V_{mdn2} are the most distant nodes from V_{max1} and V_{max2} respectively. Fig. 2 *upper* shows the situation before the insertion of new cells. Fig. 2 *lower* shows the situation after the insertion of new cells. The first new cell V_{new1} is inserted between V_{max1} and V_{mdn1}, the second V_{new2} is inserted between V_{max2} and V_{mdn2}.

B. TGCS Learning Algorithm

For 2-dimensions, TGCS performs the Delaunay triangulation of input vector space according to an unknown input vector density distribution $p(x)$. Unsupervised training of TGCS takes place by competitive Hebbian learning or WTA mode which can be described as follows:

1) Start with a random triangle of 3 connected cells.
2) Load an input pattern $\boldsymbol{x}$ chosen randomly from $p(x)$.
3) Calculate the Euclidean distance between $\boldsymbol{x}$ and all cells.
4) Find the cell with the shortest distance (i.e., "winner").
5) Accumulate the squared distance to input vector $\boldsymbol{x}$ in a local counter variable $\boldsymbol{E}_x$.
6) If the counter (or the error measure, i.e., the resource variable) exceeds a threshold:
   (a) split the edge to the first farthest topological neighbor and insert the first new cell at the halfway;
   (b) split the edge to the second farthest topological neighbor and insert the second new cell at this halfway.
7) If the counter did not change for a certain time, remove a cell together with its connections.
8) Move the winner cell and its topological neighbors towards the input pattern $\boldsymbol{x}$.
9) Iterate (2) – (8) until some performance measure has met or a predefined maximum number of grown cells is arrived.

GCS are based on an unsupervised algorithm, but can be extended to supervised learning, i.e., Supervised GCS (SGCS) [3]. This network combines two families of networks, SOFM and Radial Basis Function networks, with the ability to grow during training process. It consists of three layers, including one hidden layer. The cells of the hidden layer have a Gaussian activation function. The weight vector of each cell defines the center of the Gaussian function and an additional parameter defines its size. In addition, the hidden layer forms a topological structure defined through hyper- tetrahedrons, which is a special type of neighborhood definition of SOFM. The neighborhood also defines the standard deviation of the Gaussian activation function by using the mean distance between a cell and its neighbors. The output of the network is simply computed by a weighted sum of the cell activations in the hidden layer.

Same mechanism in SGCS can be applied to TGCS. The training algorithm of the supervised TGCS is based on three parts: the first part is concerned with the adaptation of the synaptic weights from the input to the hidden layer (input weight vectors). The second part performs insertion and deletion of cells. The classification error occurring for the training data is used to determine where to insert new cells. The third part is the adaptation of the weights from the hidden to the output layer (output weight vectors).

V. Testing Results Comparisons

We use only supervised learning for convenience comparison between different GSONN models and some traditional ones. The supervised TGCS is tested with two neural network benchmarks, the two spirals problem and the mines vs. rocks sonar classification, which are from the CMU database [17].

A. Test 1 – Two Spirals Problem

In order to solve the two spirals problem, a 2-dimensional TGCS is generated. The classification error on the training set dropped to 0 and the accumulated squared output error served for resource updating. It took TGCS 96 training epochs (CPU: 5.59 sec.) and 138 growing cells until the network achieved zero classification error. The original SGCS needed 180 epochs (CPU: about 11.68 sec.) and 145 growing cells.

Table I gives a detailed comparison between some earlier methods and the new approach for the two-spiral problem. As it can be seen, the number of epochs and CPU time required by the TGCS is about twice smaller than that required by the original SGCS or DCS-GCS.

TABLE I
EPOCHS FOR SUPERVISED LEARNING OF THE TWO-SPIRALS PROBLEM. THE FIRST PART OF THE TABLE IS TAKEN FROM J.BRUSKE[10]. THE TIME UNIT IS SECOND

Algorithm	Epochs	CPU Time	Grown Cells
Standard BP	20000		
Cross Entropy BP	10000		
Cascade-Correlation	1700		
DCS-GCS	177		
SGCS	180	10.24	145
TGCS	96	5.59	138

B. Test 2 – Mines/Rocks Separation with Sonar

This benchmark is used to test TGCS. In the experiment the accumulated squared output error served as resource for weight updating. To achieve 92.31% classification rate on the test samples, the TGCS network needs only 78 epochs and 147 grown cells. At the same time, the classification error on the training set dropped to 0. By comparison, after 140 epochs the SGCS reached 90.38% accuracy for test samples with 141 grown cells. At the same time, the classification error on the training set fell to 3.85% (it is 0 by TGCS). The supervised DCS-GCS reached an accuracy of 91.4% in the test samples after 95 epochs of training. Table 2 shows the difference between SGCS and TGCS for this problem.

Although the double growing cell mechanism can reduce the required number of learning epochs, it can also lead to a

more complex network structure. Simulation results on some more neural network benchmarks indicate that, for many datasets, when the number of new cells (which are inserted at the same time in each insertion step) is two or three, the total performance of networks is at the best, but that increasing the number of new cells beyond three has very little benefit and sometimes degrades performance.

TABLE II
COMPARISON BETWEEN TGCS AND SGCS FOR SONAR MINE/ROCK SEPARATION PROBLEM. THE FIRST PART OF THE TABLE IS TAKEN FROM J.BRUSKE [10].THE TIME UNIT IS SECOND. KNN--K-NEAREST NEIGHBOR CLASSIFIER, CR--CLASSIFICATION RATE

Algorithm	Epochs	CPU Time	Test CR
KNN			82.7%
MLP			90.4%
DCS-GCS	95		91.4%
SGCS	140	29.45	90.38%
TGCS	78	17.36	92.31%

VI. CONCLUSIONS

The main advantage of a GSONN is that it can automatically find a network structure and size suitable for a given input pattern through cell insertions. GSONN can dynamically produce a self-organizing network with smaller size and better generalization performance and rapid convergence capacity. The main characteristics of GSONN can be summarized as: (a) The network structure is determined automatically from the input data; (b) The network size need not to be predefined. Instead, the growth process can be continued until a performance criterion is met; (c) The insertion of new units can be influenced such that the generated network estimates the probability density of the input signals as well as minimizing the quantization error. The supervised learning formed by GSONN also has some advantages over the traditional neural network models: (a) The number, diameter and position of RBF units are determined automatically through a growth process; (b) The temporal classification error can be used to determine where to insert new RBF units; (c) The networks are relatively small and generalize very well.

The final topology of GSONN is a function of the growing algorithm, i.e., the learning rate parameters, the growing way and the number of new units. An interesting goal would be a model with no parameters except the properties of the desired classifier. GSONN combined with evolutionary computation and fuzzy sets would be an exciting research direction. Based on GSONN, we are now focusing on a hybrid soft computing platform for intelligence oil & gas exploration data processing and reservoir modeling.

REFERENCES

[1] B. Fritzke, Some competitive learning methods. http://www.neuroinformatik..ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/
[2] T. Kohonen, Self-Organizing Maps. Springer, Berlin, Heidelberg, 2001. (Third Extended Edition).
[3] Fritzke, Growing cell structures – a self-organizing network for unsupervised and supervised learning. Neural Networks, 7(9):1441–1460, 1994.
[4] A. Zell, M. Schmalzl, Dynamic LVQ—A fast neural net learning algorithm. In Proc. ICANN'94, International Conference on Artificial Neural Networks, volume II, pages 1095–1098, London, UK, 1994.
[5] J. Goppert, W. Rosenstiel, Interpolation in SOM: Improved generalization by iterative methods. In Proc. ICANN'95, International Conference on Artificial Neural Networks, pages 69–74, France, 1995.
[6] A. Zell, H. Bayer, H., Bauknecht, Similarity analysis of molecules with self-organizing surfaces—an extension of the self-organizing map. In Proc. ICNN'94, International Conference on Neural Networks, pages 719–724, Piscataway, NJ, 1994. IEEE Service Center.
[7] D. Deng, N. Kasabov, On-line pattern analysis by evolving self-organizing maps Neurocomputing 51: 87-103 (2003)
[8] J. Blackmore, Visualizing high-dimensional structure with the incremental grid growing neural network. Technical Report AI95-238, University of Texas, Austin, August 1, 1995.
[9] M. Dittenbach, F. Merkl, A. Rauber, The growing hierarchical self-organizing map. In S. Amari, C. L. Giles, M. Gori, and V. Puri, editors, Proc of the International Joint Conference on Neural Networks (IJCNN 2000), volume VI, pages 15 – 19, Como, Italy, July 24. – 27. 2000. IEEE Computer Society.
[10] J. Bruske, G. Sommer, Dynamic cell structure learns perfectly topology preserving map. Neural Computation, 7(4):845–865, 1995.
[11] T. Martinetz, Competitive Hebbian learning rule forms perfectly topology preserving maps. In Stan Gielen and Bert Kappen, editors, Proc. ICANN'93, Int. Conf. on Artificial Neural Networks, pages 427–434, London, UK, 1993. Springer.
[12] V. Hodge, J. Austin, Hierarchical growing cell structures: TreeGCS. In IEEE TKDE Special Issue on Connectionist Models for Learning in Structured Domains.
[13] N. Vlassis, A. Dimopoulos, G. Papakonstantinou, The probabilistic growing cell structures algorithm. Lecture Notes in Computer Science, 1327:P649, 1997.
[14] J.Pakkanen, J. Iivarinen, E. Oja, The evolving tree -- a novel self-organizing network for data analysis. Neural Processing Letters, 20(3):199{211, 2004.
[15] Y. Wang, C. Yang, K. Mathee, G. Narasimhan, Clustering using Adaptive Self-Organizing Maps (ASOM) and Applications. Proceedings of International Workshop on Bioinformatics Research and Applications, p944-951 Atlanta, Georgia, May 2005.
[16] R. Freeman and H. Yin, "Adaptive Topological Tree Structure (ATTS) for document organization and visualization," Neural Networks, Vol. 17, pp. 1255-1271, 2004.
[17] M. White, S. Fahlman, CMU repository of neural network benchmarks. Technical report, The Carnegie Mellon University, 1997.