Journal of Computing and Security

January 2014, Volume 1, Number 1 (pp. 47–59)

http://www.jcomsec.org

A Growing Hierarchical Approach to Batch Linear Manifold Topographic Map Formation

Peyman Adibi [a,*]

[a] *Department of Computer Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 15 December 2012
Revised: 23 June 2013
Accepted: 17 July 2013
Published Online: 20 December 2013

Keywords:
Unsupervised Learning, Linear Manifold Topographic Map, Growing, Hierarchical

ABSTRACT

The linear manifold topographic map (LMTM) is a model for unsupervised learning of multiple low-dimensional linear manifolds from a set of data samples in a topology-preserving map. Several limitations exist in LMTM and many other topographic maps that relate to their fixed topology and fixed number of representation elements or neurons. In this paper, a growing hierarchical structure is proposed for LMTM, to remove these limitations, and to be able to take advantage of the possible hierarchical nature of the datasets. The attempt is made to avoid several existing problems in the similar hierarchical and growing structures, through this proposed algorithm. Experimental results indicate the proper performance of the model on a synthesized and two real-world problems. In the first experiment, a complex manifold constituted of a circle, a line, and a sinusoidal part is properly represented by the proposed model. In the second and third sets of the experiments, the superiority of the proposed model is shown in comparison with the other related methods on an image compression problem and a handwritten digit recognition application.

1 Introduction

Linear Manifold Topographic Map (LMTM) [1–4] is a neural model for learning regional (or local) linear approximations of the underlying data manifolds, through a topology-preserving lattice. The learning process is performed in an unsupervised manner and obtains a piecewise linear representation of the data manifold. The topology preservation property makes the model applicable to some problems which require this feature, such as data exploratory and data visualization applications [5].

The LMTM, due to linearity of its local manifolds, is simpler and more reliable in real-world problems, than some global nonlinear manifold learning approaches, such as principal curves and surfaces [6, 7], nonlinear principal component analysis (NLPCA) [8], kernel principal component analysis (KPCA) [9], and various spectral methods [10] (see[2] for more details). It is also more flexible and precise than the methods which learn only one global linear manifold for data representation, such as principal component analysis (PCA) [11] and independent component analysis (ICA) [12]. The regional dimensionality estimation capability is also proposed for this model [4], which can improve its representation quality by increasing the precision or decreasing the amount of the required memory. Taking advantage of such ability is not possible for the linear or nonlinear global approaches, containing the above mentioned methods.

A limitation of LMTM and many other topographic maps derived from the well-known self-organizing map (SOM) network [13], relates to their fixed and prede-

* Corresponding author.
Email address: adibi@eng.ui.ac.ir (P. Adibi)
ISSN: 2322-4460 © 2014 JComSec. All rights reserved.

fined topology (that is usually a rectangular or hexagonal two-dimensional topology). From the representation point of view, such fixed topologies may decrease the representation accuracy of the network by preventing the convergence of the map to the global optimum solution. Moreover, the fixed map structure does not reflect the data distribution in a direct and clear manner. In these maps, a neuron has the same distance to all its neighbors, independent of the closeness of their corresponding data clusters. Thus, additional processes, such as the so-called U-matrix method [14], are required to make an indirect visual representation of data distribution. From the cluster analysis point of view, the fixed structure of SOM does not reflect the proximity of the clusters represented by the neurons. To resolve such limitations, the maps with more flexible topologies, such as double SOM [15], are also proposed.

Another limitation of LMTM and many other fixed-structure neural maps, results from the fixed number of their representation elements or neurons. An initial choice for this number may not match the requirements of the problem. Actually, without prior knowledge about the data distribution, it is hard to define a predetermined size for the network to achieve satisfying performance. Several proposed growing versions of the self-organizing networks [16–20], try to fix this problem.

On the other hand, there are many real-world applications which have a hierarchical nature. For example, the problem of arrangement, exploration, and search in a massive collection of information, can be handled more efficiently by a hierarchical approach. Also, using the hierarchical models, a complex problem can be divided into several simple sub-problems, which can be efficiently solved. These partial solutions can be combined to make a solution for the original problem [21]. Another advantage of a hierarchical learning method is the ability of finding a multi-resolution data representation. The hierarchical [22] and tree-structured [23] versions of the self-organizing map are developed based on such motivations.

Many researchers considered the growing and hierarchical map structures together [24–29]. In this manner, the hierarchical data are supported, and the number of required neurons is obtained based on the characteristics of the problem. These methods usually follow a top-down or divisive hierarchical clustering [29] approach, i.e. they begin from a single cluster containing the complete data set, and divide it into smaller sub-clusters in a hierarchical manner, to reach a desired representation accuracy at the end.

In this article, a growing hierarchical structure is proposed for LMTM, to resolve the limitation of the fixed topology and the fixed number of neurons of the map, and provide the opportunity to take advantage of the possible hierarchical nature of datasets. As will be observed, in development of the learning algorithm of this model, the attempt is made to avoid the problems existing in the similar hierarchical and growing structures. In Section 2, an overview of several related models is presented. In Section 3, the batch LMTM network is briefly discussed. Section 4 introduces the proposed network, and details of its learning and growing algorithm. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Several related models which are inspiring in development of the proposed network architecture are briefly studied here. These contain the tree-structured and two growing hierarchical variants of SOM and a flexible-topology version of this network. Moreover, several alternate hierarchical topographic map formation approaches are reviewed as well.

The tree structured self-organizing map (TS-SOM) [23] is a fast implementation of SOM. The speed of the SOM algorithm is increased by a tree search in finding the winning neuron and using a limited version of the search and update steps [23]. A balanced tree with a predefined number of layers is used to learn a multi-resolution representation of data. Each layer is a SOM with 1-D or 2-D lattice. The lattice dimension and topology is the same for all layers. Each neuron of a map has a fixed number of children in the next layer of the tree (which is 2 for 1-D and 4 for 2-D topologies). The TS-SOM network does not learn a hierarchical representation of the data. Since the tree is balanced and complete, many neurons, especially in the last layers, may be redundant from the representation point of view.

The growing hierarchical self-organizing map (GHSOM) [30] tries to learn and represent the hierarchical relationships between data samples and clusters. Beginning from the first layer, contained in a single-neuron SOM, the growing takes place in depth of the hierarchy. This occurs by selecting a neuron of the current layer with a poor representation quality, and inserting a small SOM as the child of this neuron into the next layer. Each SOM has a rectangular-topology. A new generated SOM also grows in its layer by insertion of rows or columns of neurons between neighboring neurons with the most different weight vectors. The growing in depth and in width of each layer is terminated when some user-defined thresholds of representation accuracy are met.

Unbalanced hierarchical growing in GHSOM decreases the loss of processing units (or redundant neurons) as mentioned for TS-SOM. High dependence of GHSOM algorithm to the user defined thresholds is a weakness of this model. Also, the problem of redundant neurons still exists in this model, when a row or a column of neurons is inserted in the map. It is clear that in an inserted row (or column) all neurons are not essential to reach the desired representation quality. This problem is a result of fixed topology selected for the maps in this model.

In the growing hierarchical tree SOM (GHTSOM) network [29], the growth is done only in depth. Simple maps with small number of neurons (triangle SOMs) are added in each growing step. The algorithm contains two steps: training and clustering. The training process starts with one triangle SOM as the root of the tree and each neuron grows by adding one triangle SOM to the next layer. The triangular topologies are only valid in the training step. The clustering process ignores these topological connections and creates special connections, called class links, between the neurons which represent the same clusters.

Most of the problems with the above mentioned methods are resolved in the GHTSOM network. A proper hierarchical data representation is achievable by this model. The network works almost free of the user-defined parameters. Adding small SOMs helps to prevent the loss of neurons, as well. But still one of the three neurons of an added triangle SOM may become redundant during the algorithm. Thus, a neuron deletion mechanism is proposed to remove this redundancy [29]. Also, as mentioned before, the role of the fixed topologies selected for the triangle SOMs is somehow ignored during the algorithm. This may be the result of the problems emerged from the fixed topologies, as observed in GHSOM.

To avoid the problems arising from the fixed-topology maps, several noticeable flexible topologies are proposed for SOM like double SOM (DSOM) [15] and its adaptive version (ADSOM) [31]. In these models, instead of having fixed positions on a lattice, each neuron has a dynamic (usually two-dimensional) coordinate in the topological space, called position vectors, which are updated during the adaptation of the weight vectors. The on-line adaptation algorithms in these models, try to adjust the distances between the position vectors of the neurons, such that they become proportional to the corresponding distances between their weight vectors. After convergence, the resulting map yields a more direct visualization of the cluster centers, compared to the standard SOM.

Two extensions of the self-organizing map are proposed to deal with tree-structured data, namely SOM for structured data (SOM-SD) [32] and merge SOM (MSOM) [33]. The SOM-SD is designed for the processing of labeled directed acyclic graphs, throughout a recurrent neural architecture. The label of each node in the tree is processed within the context presented by its child subtrees. Each neuron encodes the context using a number of additional vectors containing the indices of the winner neurons for the subtrees of that neuron. The winning neuron index is recursively computed in a bottom-up manner from the leaves to the root of the tree. An empty context is assigned to the leaves, since they do not have any child subtrees. The MSOM network is designed for unsupervised sequence processing. The context vector in this network combines the sequence history using the merged property of the winning neuron. This merged property, contains the previous winners weight vector and the context vector computed for the previous element of the sequence. This network uses a free and dynamic lattice topology as well.

Several other hierarchical topographic maps are developed in the context of generative topographic map (GTM) [34], a probabilistic variant of the self-organizing map. In a model called GTM-HMTM [35], the GTM is modified by replacing the centers of its latent space (which are the counterparts of the SOM neurons) with the hidden Markov tree models (HMTM) [36], and then applied for visualization of tree-structured data. In this model, every tree is considered as an atomic node in the latent space and HMTM generates the trees in a top-down manner, i.e. from roots to the leaves. Top-down generation of the trees means that a node is evaluated in its parent context; therefore, a parent node captures little information about the co-occurrence of particular substructures in its child trees. Two kernelized versions of GTM are also proposed for graph data [37, 38]. In these models, the Euclidean distance in the Gaussian mixture of the GTM is replaced by a kernel-based metric. The kernel GTM (KGTM) [38] is defined in terms of graph similarity, while the relational GTM (RGTM) [37] uses a matrix of graph dissimilarity.

In kernel-based models and the GTM-HMTM, a structure is treated as an atomic i.i.d. observation. As a result, a topographic mapping is only associated to the whole structure. On the other hand, recursive models focus on each single node composing the graph, allowing consideration of their context in their topographic mapping. This property is called compositionality, which means the ability of exploiting the modularity of data by first considering substructures, and then composing the contextual information of the main structures. Recently, the generative topographic mapping for structured data (GTM-SD) [39], is proposed as a compositional generative model for topographic mapping of tree-structured data. GTM-SD applies a bottom-up hidden-tree Markov model [40] to develop a recursive topographic mapping of hierarchical information. The model efficiently uses the contextual information from substructures. Experiments indicate that the projection space of GTM-SD yields a fine grained discrimination of the sample structures.

3 Batch linear manifold topographic map

3.1 Preliminary definitions

There are K neurons arranged in a 2-dimensional topographic map L. To each neuron r of the map, a linear manifold, determined by a mean vector $\boldsymbol{m}_r$ and d orthonormal basis vectors $\{\boldsymbol{b}_r^1, \cdots, \boldsymbol{b}_r^d\}$, is assigned. The set of learning subjects thus contains the mean and the basis vectors of all the map neurons. A neighborhood function or lateral interaction h_{rs} is defined between each of the two neurons r and s of the map, which is a decreasing function of the lattice distance d_{rs} between the two neurons.

The n-dimensional data points are assumed to regionally located on multiple d-dimensional linear manifolds, with $d < n$. This assumption is approximately true when the data are generated by an underlying process with lower degrees of freedom (d) than the input dimension (n). For an input vector $\mathbf{x}^\mu$ and a neuron r, a vector $\boldsymbol{\varphi}_r^\mu$ is defined as

$$\boldsymbol{\varphi}_r^\mu = \mathbf{x}^\mu - \boldsymbol{m}_r \tag{1}$$

which can be decomposed into two orthogonal vectors $\hat{\boldsymbol{\varphi}}_r^\mu$ and $\tilde{\boldsymbol{\varphi}}_r^\mu$ by projecting onto the linear manifold as

$$\hat{\boldsymbol{\varphi}}_r^\mu = \sum_{k=1}^{d} (\boldsymbol{\varphi}_r^{\mu T} \boldsymbol{b}_r^k)\boldsymbol{b}_r^k, \quad \tilde{\boldsymbol{\varphi}}_r^\mu = \boldsymbol{\varphi}_r^\mu - \sum_{k=1}^{d} (\boldsymbol{\varphi}_r^{\mu T} \boldsymbol{b}_r^k)\boldsymbol{b}_r^k \tag{2}$$

The two on- and off-manifold distances of $\mathbf{x}^\mu$ with respect to the manifold of neuron r are then defined as

$$\hat{r}_r^\mu = \|\hat{\boldsymbol{\varphi}}_r^\mu\| \text{ and } \tilde{r}_r^\mu = \|\tilde{\boldsymbol{\varphi}}_r^\mu\|, \tag{3}$$

where, $\|\|$ is the Euclidean norm of a vector.

3.2 Free energy functional

The distance of the input sample $\mathbf{x}^\mu$ from the representation medium of neuron r is defined as

$$D(\mathbf{x}^\mu, \boldsymbol{\theta}_r) = \frac{1}{2}\left(\alpha_r^2 \hat{r}_r^{\mu^2} + \tilde{r}_r^{\mu^2}\right), \tag{4}$$

where, $\boldsymbol{\theta}_r$ is the set of learning subjects of neuron r, and $0 \le \alpha_r \le 1$ is a parameter which adjusts the mutual importance of $\hat{r}_r^\mu$ and $\tilde{r}_r^\mu$. If $\alpha_r = 0$, the distance to the mean vector (on-manifold distance) is not considered which diminishes the regional representation nature of the neuron r. On the other hand, if $\alpha_r = 1$, the effect of the basis vectors (off-manifold distance) are removed and the model becomes similar to a type of self-organizing map presented in [41]. Thus, typically we must have $0 < \alpha_r < 1$. The value of α_r in this range indicates that the role of the off-manifold distance, as expected from a manifold learning method, is more important than that of the on-manifold distance. Let p_r^μ be the probability that the input sample $\mathbf{x}^\mu$ is assigned to neuron r, with the constraints $p_r^\mu \ge 0$ and $\sum_r p_r^\mu = 1$. Let the lateral interaction strength h_{rs} (the neighborhood function) denote a confusion measure that is proportional to the probability that the input sample $\mathbf{x}^\mu$, which is considered to be represented by neuron r, is represented by neuron s instead.

Now we define the representation error as

$$F_{rep}(\boldsymbol{P}, \boldsymbol{\theta}) = \sum_\mu \sum_r p_r^\mu \sum_s h_{rs} D(\mathbf{x}^\mu, \boldsymbol{\theta}_s), \tag{5}$$

where, $\boldsymbol{\theta}$ is the set of all learning subjects $\boldsymbol{\theta}_r$ of the net and $\boldsymbol{P}$ is the set of all assignment probabilities p_r^μ. An entropy term is also considered to incorporate as many neurons as possible in learning, that is

$$F_{ent}(\boldsymbol{P}) = \sum_\mu \sum_r p_r^\mu log\left(p_r^\mu / q_r\right), \tag{6}$$

where q_r is the prior assignment probability, which is usually selected as $q_r = 1/K$ for all neurons r (K is the number of neurons). Combining the representation error and the entropy term using the regularization parameter β, a free energy functional [41, 42] is defined as

$$F(\boldsymbol{P}, \boldsymbol{\theta}) = \beta F_{rep}(\boldsymbol{P}, \boldsymbol{\theta}) + F_{rep}(\boldsymbol{P}). \tag{7}$$

Parameter β in an annealing interpretation represents the inverse temperature [41]. Minimizing the free energy of equation 7, corresponds to minimizing the error function F_{rep} and maximizing the entropy of the map (that is $-F_{ent}$), simultaneously. This tries to produce the best representation quality of the map, while maintaining the activity of all neurons by forcing equality of all assignment probabilities through the entropy term.

3.3 EM algorithm

The goal is to find optimal $\boldsymbol{P}$ and $\boldsymbol{\theta}$ which minimize equation 7. This is carried out using an expectation-maximization (EM) algorithm, in which both the expectation and maximization steps minimize the free energy $F(\boldsymbol{P}, \boldsymbol{\theta})$ of equation 7 [42].

In the expectation step, equation 7 should be minimized with respect to the assignment probabilities $\boldsymbol{P}$, assuming that learning subjects $\boldsymbol{\theta}$ are fixed, which yields

$$p_r^\mu = \frac{q_r exp\left[-\beta \sum_s h_{rs} D\left(\mathbf{x}^\mu, \boldsymbol{\theta}_s\right)\right]}{\sum_t q_t exp\left[-\beta \sum_s h_{ts} D\left(\mathbf{x}^\mu, \boldsymbol{\theta}_s\right)\right]}. \qquad (8)$$

In maximization step, equation 7 is minimized with respect to the learning subjects $\boldsymbol{\theta}$, containing mean and basis vectors of the neurons, assuming fixed assignment probabilities $\boldsymbol{P}$. For each neuron s, the mean vector is found as

$$\boldsymbol{m}_s = \frac{\sum\limits_\mu \sum\limits_r p_r^\mu h_{rs} \mathbf{x}^\mu}{\sum\limits_\mu \sum\limits_r p_r^\mu h_{rs}}, \qquad (9)$$

and the basis vectors $\boldsymbol{b}_s^1, \cdots, \boldsymbol{b}_s^d$ are found to span the subspace which is spanned by the eigenvectors corresponded to the d largest eigenvalues of matrix

$$\boldsymbol{C}_s = \sum_\mu \sum_r p_r^\mu h_{rs} \boldsymbol{\varphi}_s^\mu \boldsymbol{\varphi}_s^{\mu^T}. \qquad (10)$$

In practice, the d mentioned eigenvectors are considered as the basis vectors. Derivations are given in [5].

4 The proposed growing hierarchical network

In this section we propose a new growing hierarchical LMTM network. In the proposed growing hierarchical architecture for LMTM network, we try to take advantage of the related structures, mentioned in Section 2, while avoiding their problems. An efficient hierarchical representation is realized by unbalanced growing of the proposed network in depth and adding minimal number of neurons in each growth step. The later idea resolves the problem of redundant neurons without any additional computations, such as finding and deletion of redundant neurons [29]. In continue, at first a general discussion about the network is presented in Section 4.1. Then, the details of the learning and hierarchical growing algorithm are given in Section 4.2.

4.1 Overview of the model

In the proposed model, starting from one neuron, called the root node, an unbalanced binary tree of neurons is constructed by a hierarchical growing process. Each node (neuron) of the tree has two children or no child. The nodes with no child are called the leaf nodes. The leaf nodes in each step of the algorithm constitute an LMTM with a dynamic topology, called the network of current leaves. Dynamic topology means that the position vectors of the leaf neurons are updated during the adaptation of their mean and basis vectors (see Section 4.2 for details). The root node, as an LMTM with one neuron, learns the principal d-dimensional linear manifold of the data set, and its representation error is computed. The growth step is performed by finding the leaf neuron with maximum representation error among the current leaves, and expanding it. Two neurons are generated as the children of the expended node. After expansion, the network of current leaves is updated. In this step, at first the two new generated neurons, as an LMTM with two neurons, are trained using the data samples assigned to their father. Then the assignment probabilities and the dynamic topology of the total network of current leaves are updated. The growing and updating steps are alternatively iterated until the stop condition is established.

4.2 The algorithm

Step 0. constitute layer zero: Initially, an LMTM with one neuron is trained using the total data set. This single-neuron map, indicated by , constitutes the layer number zero of the hierarchy and the root of the current tree. The representation error for the leaf nodes of the tree is computed as:

$$F_{rep}^r(L) = \sum_{\mu | \mathbf{x}^\mu \in S(L)} p_r^\mu(L) \sum_{s \in L} h_{rs}(L) D(\mathbf{x}^\mu, \boldsymbol{\theta}_s), \forall r \in L \qquad (11)$$

where L indicates the topographic map of the leaf neurons of the current tree, called the network of the current leaves. The assignment probabilities and the neighborhood functions in this network are denoted by $p_r^\mu(L)$ and $h_{rs}(L)$, representing p_r^μ and h_{rs} in the original LMTM (Section 3). The distance of sample from the neuron s, denoted by , is computed through Equation 4. The set $S(L)$ contains the samples assigned to the father of network L. In layer zero, this set is the same as the total set of the data samples (X).

Step 1. select one of the leaves and expand it: The neuron with maximum representation error in the network of current leaves is selected as $r^* = argmax_r F_{rep}^r(L)$. Expanding r^* is performed by generating a new LMTM, called M, in the next layer whose nodes are the children of r^*. The map M is noted as the expanded network. The minimal number of nodes, say 2, is considered for the expanded network. The topographic network of the leaves after expansion (the network of new leaves), denoted by L^{new}, is fabricated simply by substituting the neurons of the net M instead of their father r^* in the net L.

Step 2. update the network of new leaves:

Step 2.1. update learning subjects: The net M is trained by data set $S(M)$ containing the data samples assigned to its father r^* , that is

$$S(M) = \{\mathbf{x}^\mu \in S(L) | r^* = argmax_r p_r^\mu(L)\} \quad (12)$$

Step 2.2. update assignment probabilities: for all nodes t and training samples $\mathbf{x}^\mu$, we have

$$p_t^\mu\left(L^{new}\right) = \begin{cases} p_{t(L)}^\mu(L) \times p_{t(M)}^\mu(M) & t \text{ is a new expanded node} \\ p_{t(L)}^\mu(L) & \text{otherwise} \end{cases} \quad (13)$$

where $t(L)$ and $t(M)$ are the indices of neuron t in the nets L and M, respectively.

Step 2.3. update topology: a dynamic topology is considered, in which a coordinate $\boldsymbol{c}_s$ is assigned to each neuron s, updated as

$$\boldsymbol{c}_s\left(L^{new}\right) = \begin{cases} \boldsymbol{c}_s^{new}(L^{new}) & s \text{ is a new expanded node} \\ c_{s(L)}(L) & \text{otherwise} \end{cases} \quad (14)$$

where $s(L)$ is the index of the neuron s in the net L and $\boldsymbol{c}_s\left(L^{new}\right)$ is the new position of the neuron s in the net L^{new} , which is obtained based on an idea similar to the curvilinear component analysis (CCA) approach [43] (Equation 16) discussed in continue.

In the topographic map, the map distance between each two neurons r and s, $d_{rs} = \|\boldsymbol{c}_s\left(L^{new}\right) - \boldsymbol{c}_r\left(L^{new}\right)\|$ should reflect the distances between their assigned data samples in the input space. The later is evaluated as

$$l_{rs} = \frac{1}{(N_r + N_s)} \sum_\mu (a_r^\mu + a_s^\mu) |D^{1/2}(\mathbf{x}^\mu, \boldsymbol{\theta}_r) - D^{1/2}(\mathbf{x}^\mu, \boldsymbol{\theta}_s) \quad (15)$$

with a_r^μ and a_s^μ being the hard assignments of the sample $\mathbf{x}^\mu$ to neurons r and s, and N_r and N_s the number of data samples assigned to the neurons r and s, respectively. Now, a cost function

$$E = \frac{1}{2} \sum_r \sum_{s \neq r} (l_{rs} - d_{rs})^2 \mu(\lambda_d - d_{rs})$$

with $u(.)$ being the step function and λ_d being a locality threshold, is minimized using a stochastic gradient descent rule as follows:

$$\Delta \boldsymbol{c}_s = \eta \mu(\lambda_d - d_{rs})(l_{rs} - d_{rs}) \frac{\boldsymbol{c}_s - \boldsymbol{c}_r}{d_{rs}} \quad \forall r \in L^{new} \quad (16)$$

with η being a learning rate decreased through the learning iterations. Equation 16 is applied on all neurons s of the net M and this process is repeated for a predefined number of iterations. The coordinates $\boldsymbol{c}_s^{new}(L^{new})$ in Equation 14 will be the final values of $\boldsymbol{c}_s$s in Equation 16 at the end of iterations. Finally, the neighborhood function h_{rs} used in Equation 11, is computed as $h_{rs} = exp\left(-d_{rs}^2/(2\sigma^2)\right)$ with σ being the neighborhood width.

Step 3. check the stop condition: Considering the network L^{new} as L , the global representation error is found as

$$F_{rep}(L) = \sum_{rinL} F_{rep}^r(L) \quad (17)$$

If the global representation error is not lower than a fraction of the error in layer zero, i.e. $F_{rep}(L) \geq \alpha_0 \times F_{rep}(L_0)$, and the number of neurons in the network L is not greater than a threshold, $K(L) \leq K_{max}$, then the growth and updating steps are repeated from step 1. Otherwise, the algorithm is finished.

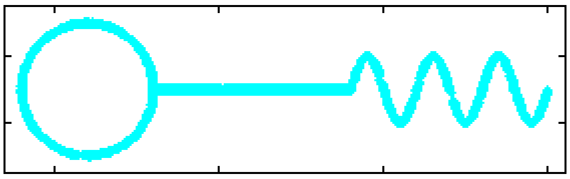
5 Experimental results

In this section, the performance of the growing hierarchical model is evaluated using a two-dimensional synthetic data set, an image compression application, and a handwritten digit recognition problem.

5.1 Sinusoid-line-circle problem

In this experiment, a circle with the radius of 2, a line with the length of 6, and three periods of a sinusoidal function with the frequency of 0.5 and the magnitude of 1, are considered. Then, 1000 data samples are randomly generated around each of these elements, such that the displacements of the samples from the original elements follow a normal distribution with zero mean and a standard deviation of 0.01. In this manner, a data set containing 3000 data samples is obtained, which can be useful for evaluating the growth property of the model (see Figure 1a). Considering the value 25 for parameter K_{max} and 10^{-6} for coefficient α_0 in step 3 of the algorithm in Section 4.2, the growing continues up to a tree with 25 leaf nodes. The learning subjects of the network of current leaves at the end of the algorithm is presented in Figure 1b. It is observed that the proposed growing hierarchical model yields an acceptable piecewise linear representation for this complex nonlinear manifold.

The model parameters are selected as $\alpha_r = 0.8$ for all neurons r, $\beta = 20$, and neighborhood width as $\sigma = 10^{-10}$. Since the goal is to demonstrate the training and growing part of the algorithm, selecting such small value for the neighborhood width makes the part of topology adaptation of the algorithm independent of the part of training and growing. The linear manifold dimension d are selected to be 1, and the number of iterations of the EM algorithm is 50. The parameter values are selected based on a trial and error process.


(a) Data distribution

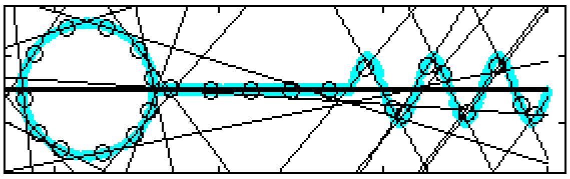
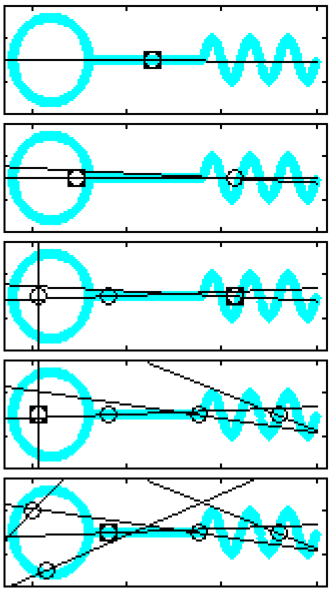
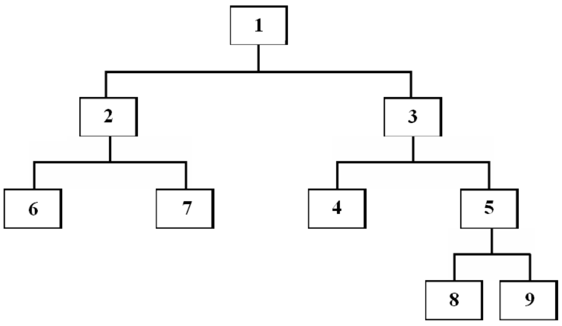

(b) Mean vectors (small circles) and 1-dimensional linear manifolds (lines)

Figure 1. Sinusoid-line-circle distribution and the representation found by the proposed model.


(a) The network of leaves in four growing steps



(b) The tree structure of the model after four growing steps

Figure 2. The representation found by the model in the first four growing steps, and the corresponding constructed tree.

The growing steps of the model are shown in Figure 2a illustrates the network of leaves in four growing steps. In each step, the neuron with maximum representation error (shown differently) is selected and two children are generated for it in the next layer. Figure 2b illustrates the tree obtained after four growing steps. The tree structure can be interpreted as follows: the root (neuron 1) corresponds to the global mean of the distribution. In the next level, the model detects the sinusoid and circle structures by neurons 2 and 3, respectively. Then, the model starts to represent these structures more precisely. This is performed by generating neurons 4 and 5 under neuron 3 and neurons 6 and 7 under neuron 2, respectively. As neuron 5 has the most representation error in this level of the tree, two children are generated under it in the next level (neurons 8 and 9) to represent the circle structure more accurately.

5.2 Image Compression

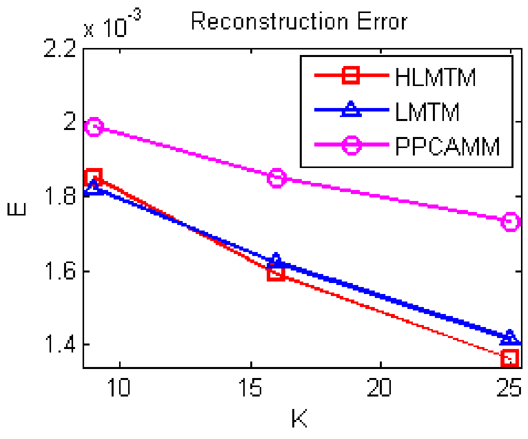
In this section, the performance of the proposed model is evaluated in a real-world image compression application. The benchmark image baboon is used for this purpose. All 88 nonoverlapping blocks constitutes a data set of 4096 samples in a 64-dimensional space. The data set is used to train the proposed growing hierarchical model, the original batch LMTM, and the probabilistic PCA mixture model (PPCAMM) [44]. The models are used with 9, 16, and 25 linear manifolds, which are arranged respectively in 33, 44, and 55 maps for the original LMTM model. For the hierarchical model, this is performed by considering values of 9, 16, and 25 for the parameter K_{max} and the small value of 10^{-6} for α_0 in step 3 of the hierarchical growing algorithm presented in Section 4.2. This value for α_0 allows the net to grow up to K_{max} leaf neurons and to be comparable with the two other fixed-topology networks. In general, the values of hyper-parameters α_0 and K_{max} which control the size of the final network and affect the accuracy and complexity of the model should be selected based on the problem characteristics and requirements through a proper model selection procedure. The mentioned parameter values in this example are only selected to have a fair comparison between different models. In all three models, the dimension of the linear manifolds equal 19 and the number of iterations of the EM algorithm equal 50.

The image coding and reconstruction processes are performed similar to [3]. For each model, the data point of each block of the image is projected on the closest manifold, which is considered as the best representative unit for that point. Then, the projection coefficients and the index of the best representative neuron are maintained for each block in the coded image. We used 7 bits for quantizing the projection coefficients. Reconstruction of an image block is performed by computing its projection to the manifold whose index stored in the coded image and using the stored coefficients for that block. The reconstruction quality of each model can be quantified using the normalized reconstruction error defined as:

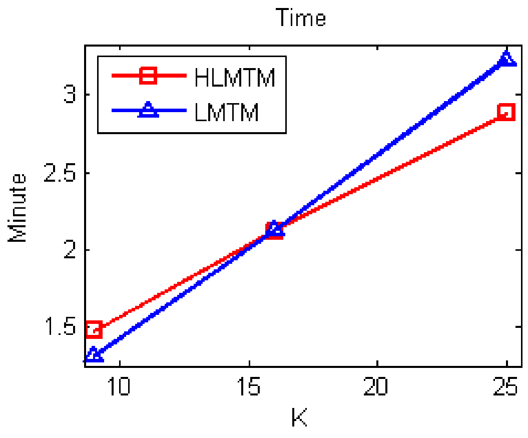
$$NRE = \frac{1}{N_p}\sum_{i=1}^{N_p}(I_i - I_i^*)^2 \qquad (18)$$

with N_p being the number of reconstructed image pixels, and I_i and I_i^* being the normalized intensities of pixel i in the original and reconstructed images, respectively. The plots of the normalized reconstruction error and the training time of different models with different network sizes, averaged in five random trials, are shown in Figure 3. The bit rates of different models with the same network size are equal to each other. It is observed that the reconstruction errors of the PPCAMM for all network sizes are considerably greater than those of the two other models (about 15 percent). For the network with smaller size, the original LMTM shows about 2 percent smaller error with about 15 percent lower training time, compared to the hierarchical model. But, in big networks, the hierarchical model starts to become better than the original one in terms of the accuracy and time. For the biggest network size in this experiment, a better performance of about 3 percent is observed from the hierarchical model, and the training time of this model is about 17 percent lower than that of the original one. This behavior can be explained as follows: in the hierarchical growing process, the proposed network tries to improve its accuracy in the regions with a higher representation error. This realizes a local improvement property for the network. But, the original LMTM tries to globally learn its manifolds through the whole data space. With larger network sizes, the flexibility of the hierarchical structure, come from its local improvement property, becomes more effective in terms of performance improvement, as focus on the higher error regions of the input space. Also, local updating process applied in the hierarchical growing algorithm obviously takes less time than the global updating of the original LMTM, which becomes more observable when the size of the network increases. The training time of the PPCAMM is much lower than the two other models, but as its accuracy was not acceptable, its training time was not reported in Figure 3, in order to the deference between the hierarchical and original LMTM training times become more observable in this figure.

The tree structure obtained from the growing and training algorithm of the proposed model with $K_{max} = 16$, in a random trial is illustrated in Figure 4. The mean vectors of the neurons are observed in this figure. It is observed that the learned tree structure is balanced, which may be a result of an almost uniform distribution of the data points in this experiment. The positions of the leaf nodes in the final topographic



(a) Reconstruction error versus the number of neurons



(b) Time (in minutes) versus the number of neurons

Figure 3. The plots of the reconstruction error (vertical axis of the upper plot) and the training time in minutes (vertical axis of the lower plot) versus the number of neurons (horisontal axes) for the original LMTM and its hierarchical version HLMTM.

map are shown in Figure 5. The topology preserving property is observed in this figure, as similar mean vectors are close to each other. It is also observed that the obtained topographic arrangement is very affective of the tree structure, such that the neurons with the same father (brothers) are posed closed to each other. In addition to the mean vectors, all the basis vectors of the leaves are shown in Figure 6.

5.3 Handwritten Digit Recognition

In this section, the proposed model is applied for recognition of the handwritten digits of the USPS data set [45]. This set contains 1100 samples for each digit 0 to 9, as 1616 images. Half of each digit set is used for training the model and the remaining half is used for test. For each digit, a network is trained using the training set of the digit. A test sample is presented to each 10 networks of the digits. The index of the network with the least representation error is considered as the recognition result. For the sake of

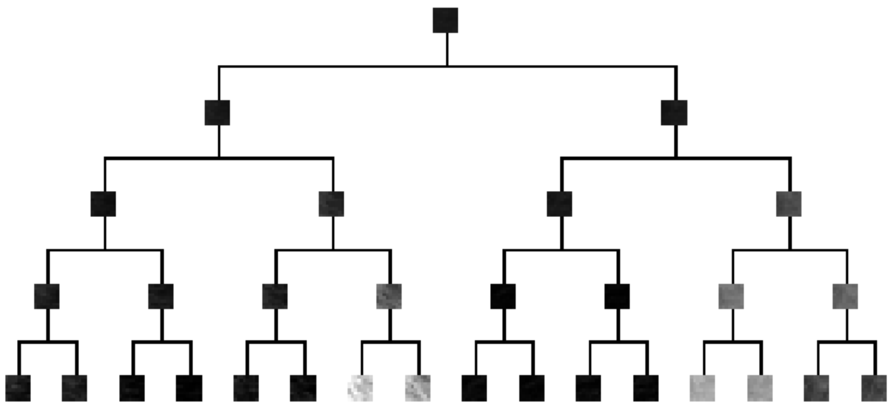
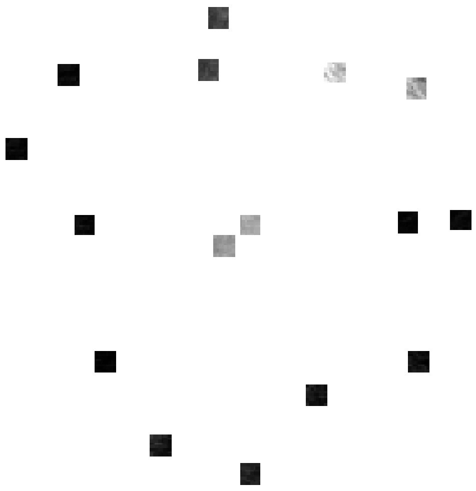
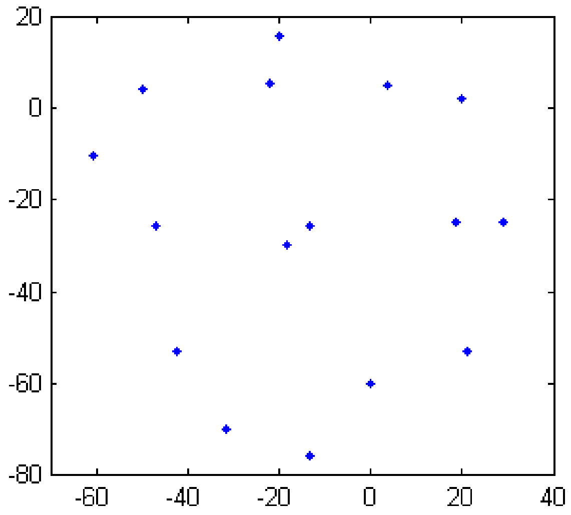


Figure 4. The tree obtained from the hierarchical growing of the proposed model with $K_{max} = 16$ in a random trial on the data set of the 'baboon' image. The nodes of the tree show the mean vectors of the corresponding neuron.



(a) The mean vectors of the neurons in their correspondin positions.



(b) The 2-dimensional positions of the neurons

Figure 5. The 2-dimensional topographic map for the network of leaves of the tree shown in Figure 4. The 2-dimensional positions of the neurons are shown using points in the lower plot. The upper plot shows the mean vectors of the neurons in the corresponding positions.

comparison, the PPCAMM and a batch version of the self-organizing map (SOM) [41] are used. The dimensionalities of the manifolds for PPCAMM and the proposed model are globally estimated, which are obtained for the training sets of digits 0 to 9, as 10, 8, 12, 13, 11, 12, 10, 9, 12, and 10, respectively [5].

Networks with different number of neurons are tested. The mean and standard deviation of correct recognition rate, in five trials with different random initializations, are presented in Table 1, for the networks with different sizes. The number of neurons for the proposed model is the number of leaf nodes, which is denoted by K_{max} in the algorithm. For the SOM network with 9, 12, and 16 neurons, the neurons are arranged in 33, 34, and 44 maps. It is observed from Table 1 that the proposed network has the best recognition rate among all other reported results in this table, which is obtained with a net size of 9. The recognition rate of SOM is considerably lower than the two other methods. This confirms the advantage of learning manifolds instead of the cluster centers in the popular clustering methods (which SOM usually is considered to be one of these methods). In addition, an increase in the network size in the manifold-based methods (HLMTM and PPCAMM) results in a minor decrease in the test accuracies, while this is not true in the clustering-based method (SOM). This behavior once more is indicative of the fact that manifold-based methods work better than the clustering-based ones. By increasing the network size, the manifold-based methods start to over-fit to the training set, which means that those are too powerful models for this (moderately small) data set. This is an expected outcome, since a manifold is a much powerful repre-

Table 1. The mean and standard deviation of the recognition rate (in percent) for the proposed method (HLMTM), PPCAMM, and SOM networks, with different sizes

Network \ Size	9	12	16
HLMTM	96.09 ± 0.202	95.56 ± 0.316	94.50 ± 0.494
PPCAMM	95.91 ± 0.137	95.63 ± 0.361	95.19 ± 0.275
SOM	91.39 ± 0.262	91.55 ± 0.188	91.99 ± 0.290

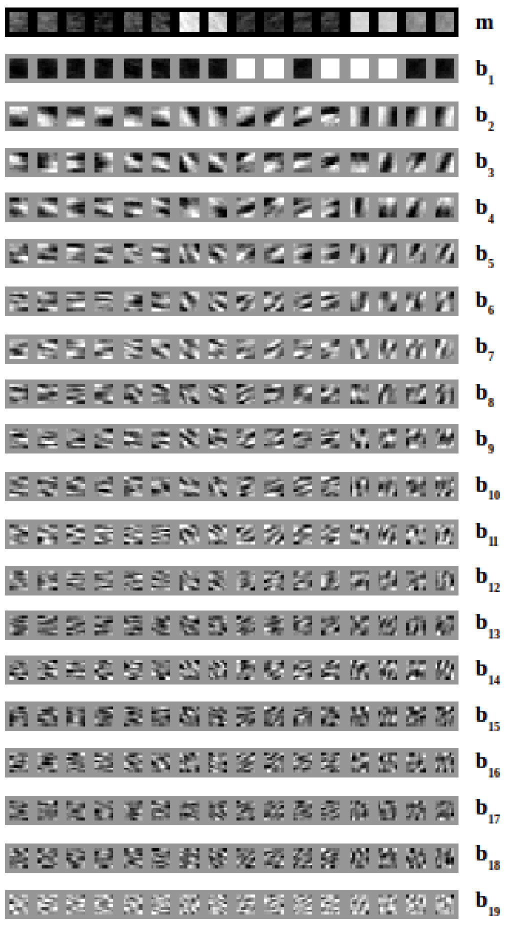
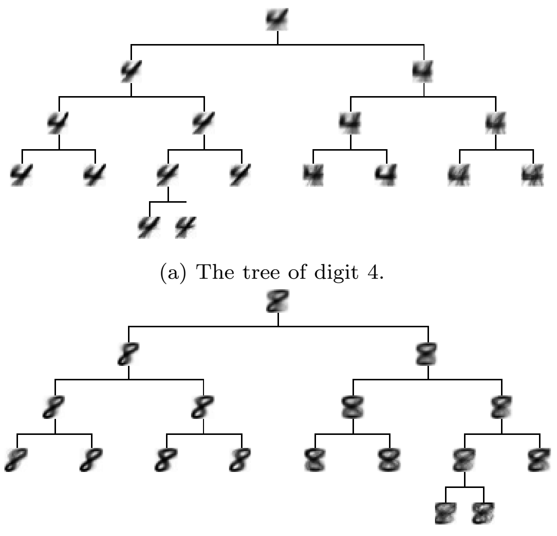


Figure 6. The mean vectors and the basis vectors of the leaves of the tree shown in Figure 4

sentation element than a cluster center.

Figure 7 shows the resulting tree for digits 4 and 8 with $K_{max} = 9$ neurons, in a random trial. The mean vectors of the neurons are shown in this figure. As expected, it is observed that in each specific level of the tree, each node is converged to one special type of the digits, and the neurons in the lower levels of the tree are specialized to represent sub-classes of the samples assigned to their ascendants in the higher levels. Again, the resulting trees are balanced since different types of each digit are almost uniformly distributed in this data set. The topographic map obtained at the end of the algorithm is shown in Figure 8, which corresponds to the leaf nodes of the tree in Figure 7. It is observed from this figure that the neurons with the same father, i.e. the siblings, are located close to each other. Since the data represented by each two siblings are similar, this closeness is also acceptable from a topology preservation point of view.



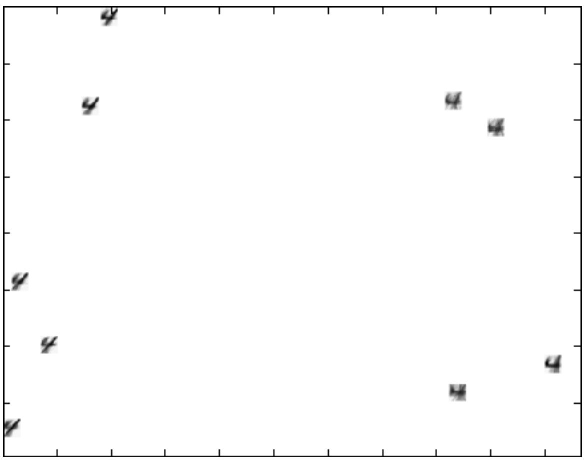
(a) The tree of digit 4.

(b) The tree of digit 8.

Figure 7. The trees obtained from the algorithm of growing and learning of the models for digits 4 and 8.

6 Conclusions

This article proposed a growing hierarchical structure for the linear manifold topographic map network, where the restrictions of the fixed topology and size of the network are removed. By considering a dynamic topology for the map and inserting minimum number of neurons in the growing process, the problem of redundant neurons are avoided; and having an unbalanced growing for the tree, the capability of hierarchical representation of the data sets is provided. The performance of the proposed model is evaluated using a synthesized data set, an image compression application, and a handwritten digit recognition problem. Comparisons with the relevant techniques show


(a) The topographic map of digit 4.

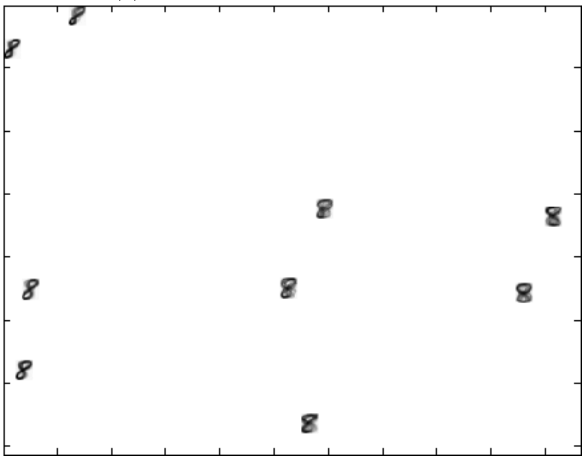

(b) The topographic map of digit 8.

Figure 8. The topographic maps of the leaf nodes of the trees shown in Figure 7.

the proper performance of the model. This approach, especially for manipulating the data sets with natural hierarchical entities can be very efficient and useful.

References

[1] Peyman Adibi and Reza Safabakhsh. Joint entropy maximization in the kernel-based linear manifold topographic map. In *IJCNN*, pages 1133–1138. IEEE, 2007.

[2] Peyman Adibi and Reza Safabakhsh. Information maximization in a linear manifold topographic map. *Neural Processing Letters*, 29(3):155–178, 2009.

[3] Peyman Adibi and Reza Safabakhsh. Linear manifold topographic map formation based on an energy function with on-line adaptation rules. *Neurocomputing*, 72(7-9):1817–1825, 2009.

[4] Peyman Adibi and Reza Safabakhsh. Batch linear manifold topographic map with regional dimensionality estimation. In *IJCNN*, pages 63–70. IEEE, 2009.

[5] Peyman Adibi. *Regional Linear Manifold Topographic Maps*. PhD thesis, Computer Engineering Dept., Amirkabir University of Technology, Tehran, Iran, 2010.

[6] Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989. doi: 10.1080/01621459.1989.10478797. URL http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1989.10478797.

[7] Robert Tibshirani. Principal curves revisited. *Statistics and Computing*, 2(4):183–190, 1992. ISSN 0960-3174. doi: 10.1007/BF01889678. URL http://dx.doi.org/10.1007/BF01889678.

[8] Juha Karhunen and Jyrki Joutsensalo. Representation and separation of signals using nonlinear {PCA} type learning. *Neural Networks*, 7(1): 113 – 127, 1994. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(94)90060-4. URL http://www.sciencedirect.com/science/article/pii/0893608094900604.

[9] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[10] Lawrence K Saul, Kilian Q Weinberger, Jihun H Ham, Fei Sha, and Daniel D Lee. *Spectral methods for dimensionality reduction*, chapter 16, pages 293–308. MIT Press, Cambridge, MA, USA, 2006.

[11] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., New York, NY, USA, 1996. ISBN 0-471-05436-4.

[12] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3): 287 – 314, 1994. ISSN 0165-1684. doi: http://dx.doi.org/10.1016/0165-1684(94)90029-9. URL http://www.sciencedirect.com/science/article/pii/0165168494900299. ¡ce:title¿Higher Order Statistics¡/ce:title¿.

[13] Teuvo Kohonen. *Self-Organizing Maps*. Springer series in information sciences, 30. Springer, 3rd edition, December 2000. ISBN 3540679219. URL http://www.worldcat.org/isbn/3540679219.

[14] A. Ultsch. Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. In Erkki Oja and Samuel Kaski, editors, *Kohonen Maps*, pages 33 – 45. Elsevier Science B.V., Amsterdam, 1999. ISBN 978-0-444-50270-4. doi: http://dx.doi.org/10.1016/B978-044450270-4/50003-6. URL http://www.sciencedirect.com/science/article/pii/B9780444502704500036.

[15] Mu-Chun Su and Hsiao-Te Chang. A new model

of self-organizing neural networks and its application in data projection. *Neural Networks, IEEE Transactions on*, 12(1):153–158, 2001. ISSN 1045-9227. doi: 10.1109/72.896805.

[16] B. Fritzke. Unsupervised clustering with growing cell structures. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume ii, pages 531–536 vol.2, 1991. doi: 10.1109/IJCNN.1991.155390.

[17] Bernd Fritzke. Kohonen feature maps and growing cell structures - a performance comparison. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, *NIPS*, pages 123–130. Morgan Kaufmann, 1992. ISBN 1-55860-274-7.

[18] J. Blackmore and R. Miikkulainen. Incremental grid growing: encoding high-dimensional structure into a two-dimensional feature map. In *Neural Networks, 1993., IEEE International Conference on*, pages 450–455 vol.1, 1993. doi: 10.1109/ICNN.1993.298599.

[19] Bernd Fritzke. Growing cell structuresa self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9): 1441 – 1460, 1994. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(94)90091-4. URL http://www.sciencedirect.com/science/article/pii/0893608094900914.

[20] Bernd Fritzke. Growing grid a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995. ISSN 1370-4621. doi: 10.1007/BF02332159. URL http://dx.doi.org/10.1007/BF02332159.

[21] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[22] Jouko Lampinen and Erkki Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(2-3): 261–272, 1992.

[23] Pasi Koikkalainen. Tree structured self-organizing maps. In Erkki Oja and Samuel Kaski, editors, *Kohonen Maps*, pages 121 – 130. Elsevier Science B.V., Amsterdam, 1999. ISBN 978-0-444-50270-4. doi: http://dx.doi.org/10.1016/B978-044450270-4/50009-7. URL http://www.sciencedirect.com/science/article/pii/B9780444502704500097.

[24] V. Burzevski and C.K. Mohan. Hierarchical growing cell structures. In *Neural Networks, 1996., IEEE International Conference on*, volume 3, pages 1658–1663 vol.3, 1996. doi: 10.1109/ICNN.1996.549149.

[25] Joaqun Dopazo and Jos Mara Carazo. Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. *Journal of Molecular Evolution*, 44(2):226–233, 1997. ISSN 0022-2844. doi: 10.1007/PL00006139. URL http://dx.doi.org/10.1007/PL00006139.

[26] Rod Adams, Kate Butchart, and Neil Davey. Hierarchical classification with a competitive evolutionary neural tree. *Neural Networks*, 12(3): 541–551, 1999.

[27] Javier Herrero, Alfonso Valencia, and Joaquín Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(1):126–136, 2001.

[28] Michael Dittenbach, Andreas Rauber, and Dieter Merkl. Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, 48(1-4):199–216, 2002.

[29] Alberto Forti and Gian Luca Foresti. Growing hierarchical tree som: An unsupervised neural network with dynamic topology. *Neural Networks*, 19 (10):1568 – 1580, 2006. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2006.02.009. URL http://www.sciencedirect.com/science/article/pii/S089360800600089X.

[30] Michael Dittenbach, Andreas Rauber, and Dieter Merkl. Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, 48(1-4):199–216, 2002.

[31] Habtom W. Ressom, Dali Wang, and Padma Natarajan. Adaptive double self-organizing maps for clustering gene expression profiles. *Neural Networks*, 16(5-6):633–640, 2003.

[32] Markus Hagenbuchner, Alessandro Sperduti, and Ah Chung Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, 2003.

[33] Marc Strickert and Barbara Hammer. Merge {SOM} for temporal data. *Neurocomputing*, 64 (0):39 – 71, 2005. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2004.11.014. URL http://www.sciencedirect.com/science/article/pii/S0925231204005107. ¡ce:title¿Trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004¡/ce:title¿.

[34] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10 (1):215–234, 1998.

[35] Nikolaos Gianniotis and Peter Tiňo. Visualization of tree-structured data through generative topographic mapping. *IEEE Transactions on Neural Networks*, 19(8):1468–1493, 2008.

[36] Matthew S. Crouse, Robert D. Nowak, and Richard G. Baraniuk. Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on Signal Processing*, 46(4):

886–902, 1998.

[37] Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. Relational generative topographic map. In *ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2010, Proceedings*, 2010.

[38] Iván Olier, Alfredo Vellido, and Jesús Giraldo. Kernel generative topographic mapping. In *ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2010, Proceedings*, 2010.

[39] Davide Bacciu, Alessio Micheli, and Alessandro Sperduti. Compositional generative mapping for tree-structured data - part ii: Topographic projection model. *IEEE Trans. Neural Netw. Learning Syst.*, 24(2):231–247, 2013.

[40] Davide Bacciu, Alessio Micheli, and Alessandro Sperduti. Compositional generative mapping for tree-structured data - part i: Bottom-up probabilistic modeling of trees. *IEEE Trans. Neural Netw. Learning Syst.*, 23(12):1987–2002, 2012.

[41] Tom Heskes. Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks*, 12(6):1299–1305, 2001.

[42] RadfordM. Neal and GeoffreyE. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In MichaelI. Jordan, editor, *Learning in Graphical Models*, volume 89 of *NATO ASI Series*, pages 355–368. Springer Netherlands, 1998. ISBN 978-94-010-6104-9. doi: 10.1007/978-94-011-5014-9_12. URL http://dx.doi.org/10.1007/978-94-011-5014-9_12.

[43] Pierre Demartines and Jeanny Hérault. Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. *IEEE Trans. Neural Netw. Learning Syst.*, 8(1): 148–154, 1997.

[44] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Comput.*, 11(2):443–482, February 1999. ISSN 0899-7667. doi: 10.1162/089976699300016728. URL http://dx.doi.org/10.1162/089976699300016728.

[45] Sam Roweis. Usps handwritten dataset. Available from http://www.cs.toronto.edu/ roweis/data.html Date of access: December 2010.

Peyman Adibi was born in Isfahan, Iran, in 1975. He received the B.S. degree in computer engineering from Isfahan University of Technology, Isfahan, Iran, in 1998, and the M.S. and Ph.D. degrees in computer engineering from Amirkabir University of Technology, Tehran, Iran, in 2001 and 2009, respectively. Since 2010, he has been with the Department of Computer Engineering, University of Isfahan, Isfahan, Iran, where he is currently an Assistant Professor. His current research interests include machine learning, soft-computing, and computer vision.