A Growing Hierarchical Approach to Batch Linear Manifold Topographic Map Formation

Peyman Adibi
Department of Computer Engineering
University of Isfahan
Isfahan, Iran
adibi@eng.ui.ac.ir

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 their 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 datasets. It is tried in the proposed algorithm to avoid the problems existing in the similar hierarchical and growing structures. Experimental results show the proper performance of the model in comparison with other related methods on a real-world image compression application.**

Keywords- unsupervised learning; linear manifold topographic map; growing; hierarchical

I. 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 for 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 for 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 that are derived from the well-known self-organizing map (SOM) network [13], relates to their fixed and predefined 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. To avoid 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. 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 to 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]. This way, the hierarchical data are supported, and the number of required neurons is found based on the characteristics of the problem. These methods usually follow a top-down or divisive hierarchical clustering [29] idea, i.e. they start from a single cluster containing total data set, and divide it to smaller sub-clusters in a hierarchical manner, to reach a desired representation accuracy.

In this paper, a growing hierarchical structure is proposed for LMTM, to remove the limitation of the fixed topology and the fixed number of neurons of the map, and to be able to take advantage of the possible hierarchical nature of datasets. As will be seen in continue, in development of the learning algorithm of this model, we try to avoid the problems existing in the similar hierarchical and growing structures.

The paper is organized as follows. 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.

II. Related Work

In this Section, several related models which are inspiring in development of the proposed network architecture are briefly studied. These contain the tree-structured and two growing hierarchical variants of SOM and a flexible-topology version of this network.

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. Starting from the first layer, contained a single-neuron SOM, the growing is performed in depth of the hierarchy. This is done 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 too different weight vectors. The growing in depth and in each layer is terminated when some user-defined thresholds of representation accuracy are reached.

Unbalanced hierarchical growing in GHSOM decreases the loss of processing units (or redundant neurons) 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 all neurons in an inserted row (or column) are not essential to reach the desired representation quality. This problem is a result of fixed type of topologies 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 SOM's) 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 of 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. Also, adding small SOM's helps to prevent the loss of neurons. 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 SOM's is somehow ignored during the algorithm. This may be the result of the problems emerged from the fixed topologies, as also observed in GHSOM.

To remove the problems arising from the fixed-topology maps, more flexible topologies proposed for SOM can be noticed. Double SOM (DSOM) [15] and its adaptive version (ADSOM) [31] are examples of these approaches. 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 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.

III. BATCH LINEAR MANIFOLD TOPOGRAPHIC MAP

A. *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 $\mathbf{m}_r$ and d orthonormal basis vectors $\{\mathbf{b}_r^1, \cdots, \mathbf{b}_r^d\}$, is assigned. The set of learning subjects thus contains the mean and 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 locate 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

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

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

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

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

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

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

B. *Free energy functional*

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

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

where θ_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 (off-manifold distance) is not considered which injures 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 [32]. 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 the role of the on-manifold distance. Let p_r^μ being 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 also the lateral interaction strength h_{rs} denoting a confusion measure that is proportional to the probability that the assigned input to neuron *r* is instead represented by neuron *s*.

Now we define the representation error as

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

where Θ is the set of all learning subjects θ_r of the net and 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

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

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 a regularization parameter β, a free energy functional [33], [32] is defined as

$$F(\mathrm{P}, \Theta) = \beta F_{rep}(\mathrm{P}, \Theta) + F_{ent}(\mathrm{P}). \tag{7}$$

Parameter β in an annealing interpretation plays the role of the inverse temperature [32].

C. *EM algorithm*

The goal is to find optimal P and Θ which minimize equation (7). This is performed using an expectation-maximization (EM) algorithm, in which both the expectation and maximization steps minimize the free energy $F(\mathrm{P}, \Theta)$ of (7) [33].

In the expectation step, equation (7) should be minimized with respect to the assignment probabilities P, assuming that learning subjects Θ are fixed, which yields

$$p_r^\mu = \frac{q_r \exp[-\beta \sum_s h_{rs} D(\mathbf{x}^\mu, \theta_s)]}{\sum_t q_t \exp[-\beta \sum_s h_{ts} D(\mathbf{x}^\mu, \theta_s)]}. \tag{8}$$

In maximization step, equation (7) is minimized with respect to the learning subjects Θ, containing mean and basis vectors of the neurons, assuming fixed assignment probabilities P. For each neuron s, the mean vector is found as

$$\mathbf{m}_s = \frac{\sum_{\mu}\sum_{r} p_r^{\mu} h_{rs} \mathbf{x}^{\mu}}{\sum_{\mu}\sum_{r} p_r^{\mu} h_{rs}}, \tag{9}$$

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

$$\mathbf{C}_s = \sum_{\mu}\sum_{r} p_r^{\mu} h_{rs} \boldsymbol{\varphi}_s^{\mu} \boldsymbol{\varphi}_s^{\mu^T} . \tag{10}$$

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

IV. The Proposed Growing Hierarchical 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 weaknesses. 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 removes 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.A. Then, the details of the learning and hierarchical growing algorithm are given in Section 4.B.

A. 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 algorithm, in addition to the learning subjects (see Section 4.B for details).

First, 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.

B. The algorithm

Step 0. Constitute layer zero: At first, an LMTM with one neuron is trained using the total data set. This single-neuron map, indicated by L_0, 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:

$$F_{rep}^{r}(L) = \sum_{\mu|\mathbf{x}^{\mu}\in S(L)} p_r^{\mu}(L) \sum_{s\in L} h_{rs}(L) D(\mathbf{x}^{\mu}, \theta_s), \quad \forall r \in L , \tag{11}$$

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 shown by $p_r^{\mu}(L)$ and $h_{rs}(L)$, which play the role of p_r^{μ} and h_{rs} in the original LMTM (Section 3). The distance of sample $\mathbf{x}^{\mu}$ from the neuron s, denoted by $D(\mathbf{x}^{\mu},\theta_s)$, is computed from 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 $r^* = \arg\max_r F_{rep}^r(L)$. 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), shown by L^{new}, is constituted 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

$$S(M) = \left\{ \mathbf{x}^{\mu} \in S(L) \mid r^* = \arg\max_r p_r^{\mu}(L) \right\}. \tag{12}$$

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

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

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 $\mathbf{c}_s$ is assigned to each neuron s, updated as

$$\mathbf{c}_s(L^{new}) = \begin{cases} \mathbf{c}_s^{new}(L^{new}) & s \text{ is a new expanded node} \\ \mathbf{c}_{s(L)}(L) & \text{otherwise} \end{cases}, \tag{14}$$

where $s(L)$ is the index of the neuron s in the net L and $\mathbf{c}_s^{new}(L^{new})$ 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 [34] (equation (16)) discussed in continue.

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

$$l_{rs} = \frac{1}{(N_r + N_s)} \sum_{\mu} (a_r^{\mu} + a_s^{\mu}) \left| D^{1/2}(\mathbf{x}^{\mu},\theta_r) - D^{1/2}(\mathbf{x}^{\mu},\theta_s) \right|, \tag{15}$$

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 $E = \frac{1}{2} \sum_r \sum_{s \neq r} (l_{rs} - d_{rs})^2 u(\lambda_d - d_{rs})$, with $u(\cdot)$ being the step function and λ_d being a locality threshold, is minimized using a stochastic gradient descent rule as follows:

$$\Delta \mathbf{c}_s = \eta u(\lambda_d - d_{rs})(l_{rs} - d_{rs}) \frac{\mathbf{c}_s - \mathbf{c}_r}{d_{rs}} \qquad \forall r \in L^{new}, \tag{16}$$

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 $\mathbf{c}_s^{new}(L^{new})$ in equation (14) will be the final values of $\mathbf{c}_s$'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

$$F_{rep}(L) = \sum_{r \in L} F_{rep}^{r}(L) \cdot \tag{17}$$

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.

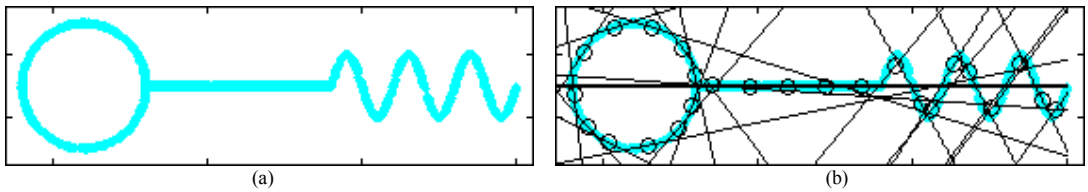
V. Experimental Results

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.

A. *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 obey a normal distribution with zero mean and a standard deviation of 0.01. In this way, a data set containing 3000 data samples, as shown in Figure 2(a), is obtained, which can be useful for evaluating the growth property of the model. Considering the value 25 for parameter $K_{\max}$ and 10^{-6} for coefficient α_0 in step 3 of the algorithm of section IV.B, the growing is continued up to a tree with 25 leaf nodes. The learning subjects of the network of current leaves at the end of the algorithm is shown in Fig. 1(b). It is observed that the proposed growing hierarchical model yields an acceptable piecewise linear representation for this complex nonlinear manifold.

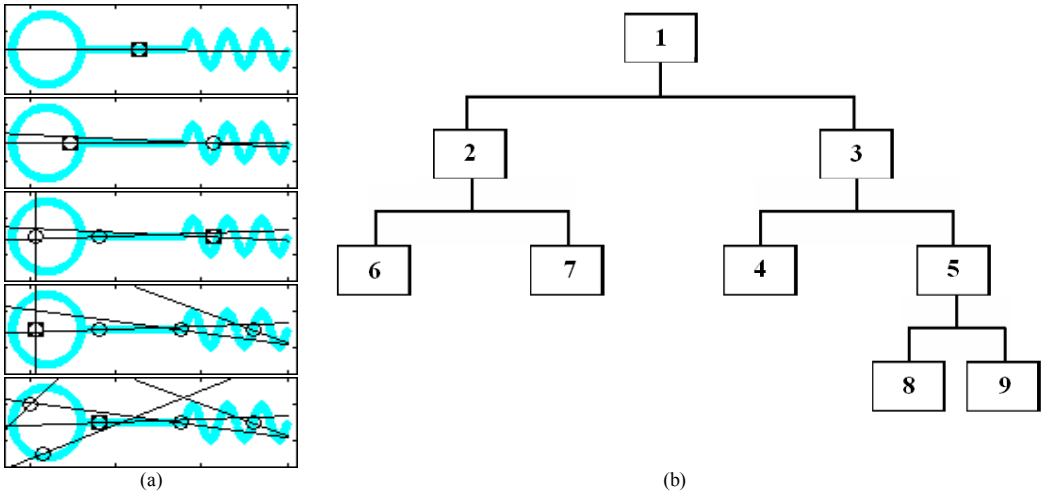
The model parameters are selected as $\alpha_r = 0.8$ for all neurons r, $\beta = 20$, and neighborhood width as $\sigma = 10^{-10}$. Since the goal is to demonstrate the training and growing part of the algorithm, selecting such small value for the neighborhood width makes the part of topology adaptation 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 equal to 50.



(a) (b)

Figure 1. (a) data distribution, (b) mean vectors (small circles) and 1-dimensional linear manifolds (lines) of the final network of leaves with 25 neurons in the proposed model.

The growing steps of the model are shown in Fig. 2. Part (a) shows 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. Part (b) of Fig. 2 shows the tree obtained after four growing steps.



(a) (b)

Figure 2. (a) The network of leaves in four growing steps, (b) The tree structure of the model after four growing steps.

B. 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 8×8 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) [35]. The models are used with 9, 16, and 25 linear manifolds, which are arranged respectively in 3×3, 4×4, and 5×5 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 algorithm of Section IV.B. In all the three models, the dimension of the linear manifolds is equal to 19 and the number of iterations of the EM algorithm is equal to 50.

The image coding and reconstruction processes are performed similar to [3], with 7 bits for quantizing the coefficients. 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 Fig. 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 greater than those of the two other models. For the network with smaller size, the original LMTM shows smaller error with a bit lower training time, compared to the hierarchical model. But, with larger network sizes, the better performance is observed from the hierarchical model, and the training time of this model is almost equal to or a bit lower than that of the original one.

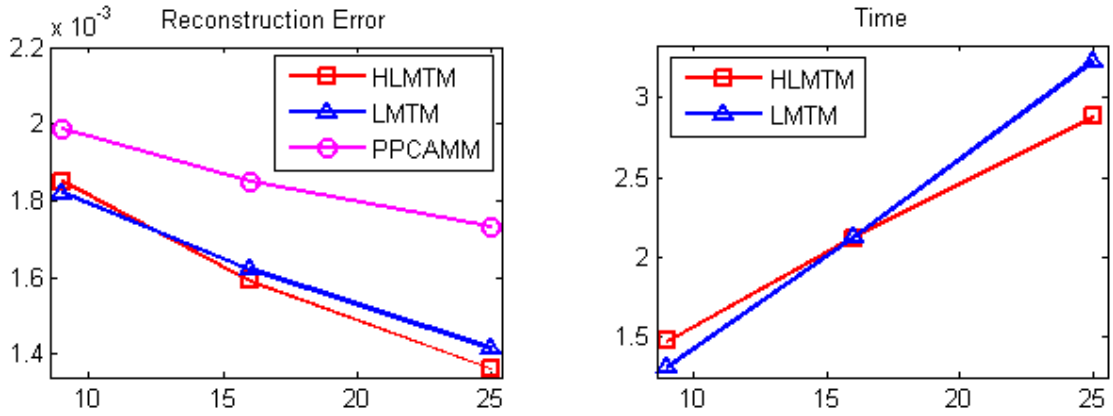


Figure 3. The plots of the reconstruction error (left) and the training time in minutes (right) versus the number of neurons (K) for the original LMTM and its hierarchical version HLMTM.

Fig. 4 shows the tree structure obtained from the growing and training algorithm of the proposed model with $K_{\max}=16$, in a random trial. The mean vectors of the neurons are observed in this figure. Fig. 5 shows the positions of the leaf nodes in the final topographic map. 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. Fig. 6, in addition to the mean vectors, shows all the basis vectors of the leaves.

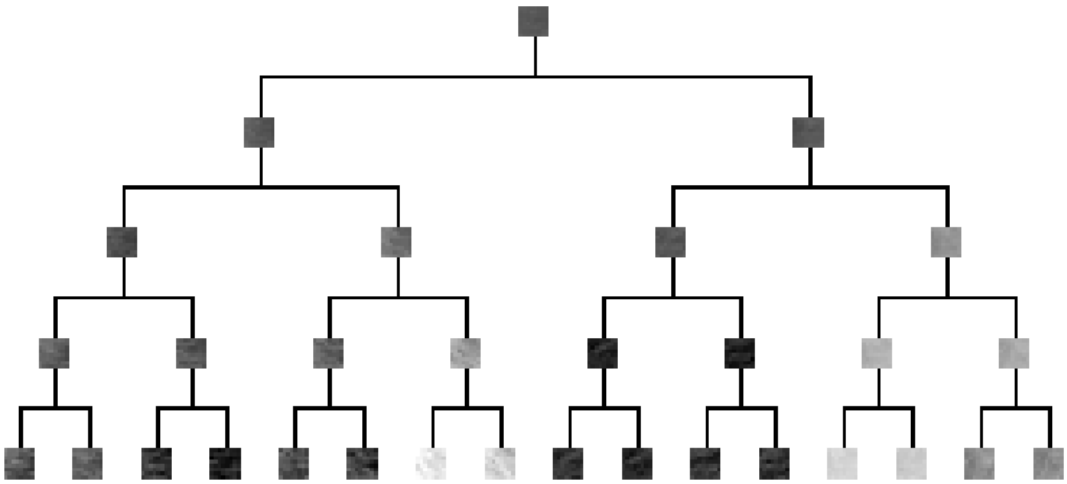


Figure 4. The tree obtained from the hierarchical growing of the proposed model with $K_{\max}=16$ in a random trial on the data set of the 'baboon' image. The nodes of the tree show the mean vectors of the corresponding neuron.

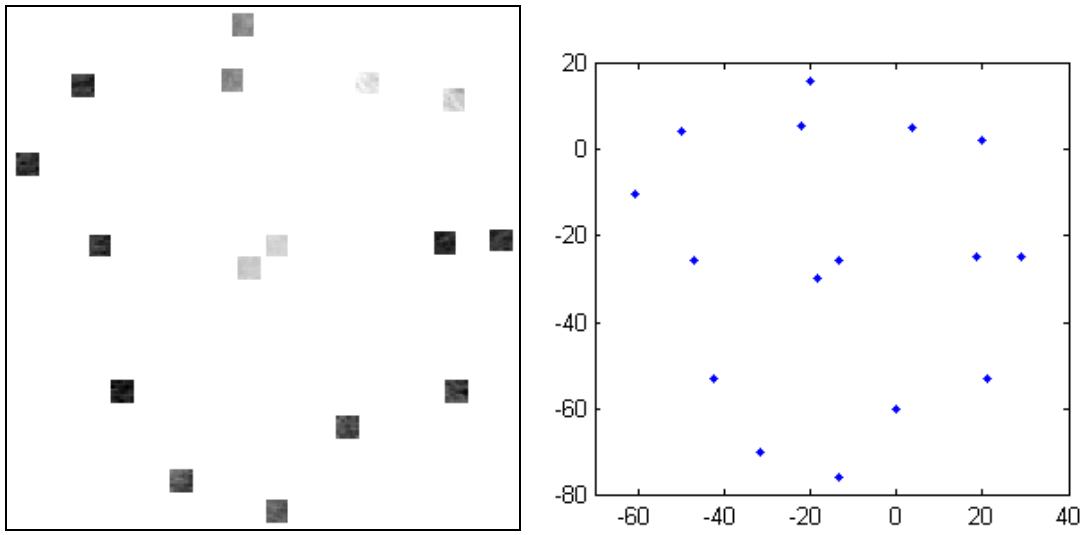


Figure 5. The 2-dimensional topographic map for the network of leaves of the tree shown in Fig. 4. The 2-dimensional positions of the neurons are shown using points in the right plot. The left plot shows the mean vectors of the neurons in the corresponding positions.

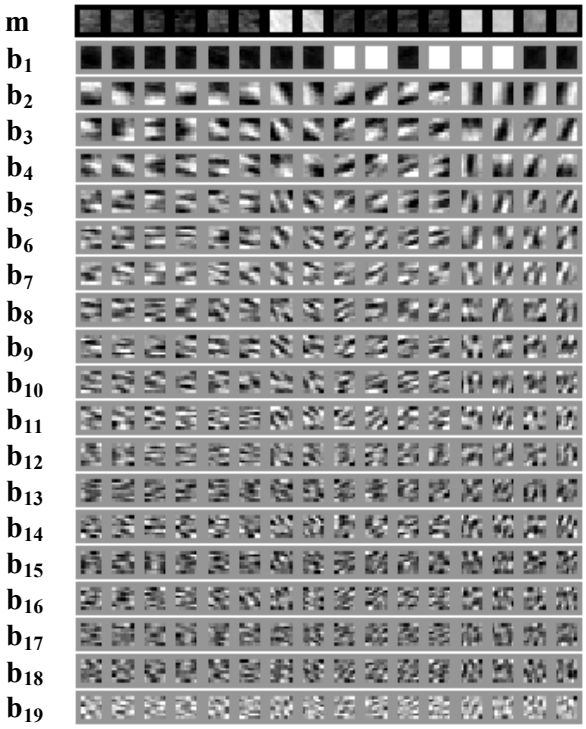


Figure 6. The mean vectors and the basis vectors of the leaves of the tree shown in Figure 4.

C. *Handwritten Digit Recognition*

In this section, the proposed model is applied for recognition of the handwritten digits of the USPS data set [36]. This set contains 1100 samples for each digit 0 to 9, as 16×16 images. One 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 comparison, the PPCAMM and a batch version of the self-organizing map (SOM) [32] are used. The dimension 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 shown 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 shown by $K_{\max}$ in the algorithm. For the SOM network with 9, 12, and 16 neurons, the neurons are arranged in 3×3, 3×4, and 4×4 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).

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

Fig. 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. The topographic map obtained at the end of the algorithm is shown in Fig. 8, which corresponds to the leaf nodes of the tree in Fig. 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.

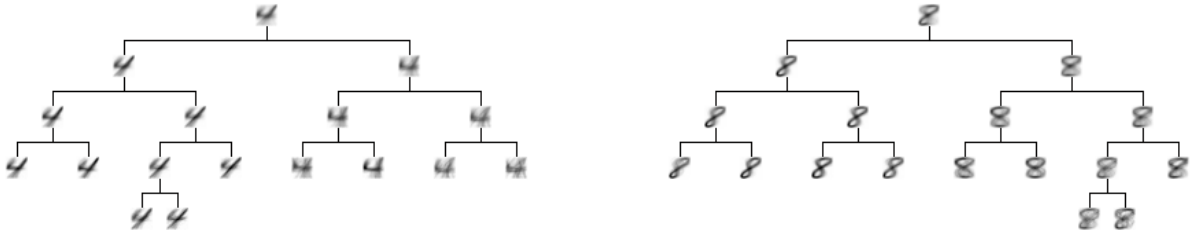


Figure 7. The trees obtained from the algorithm of growing and learning of the models for digits 4 and 8.

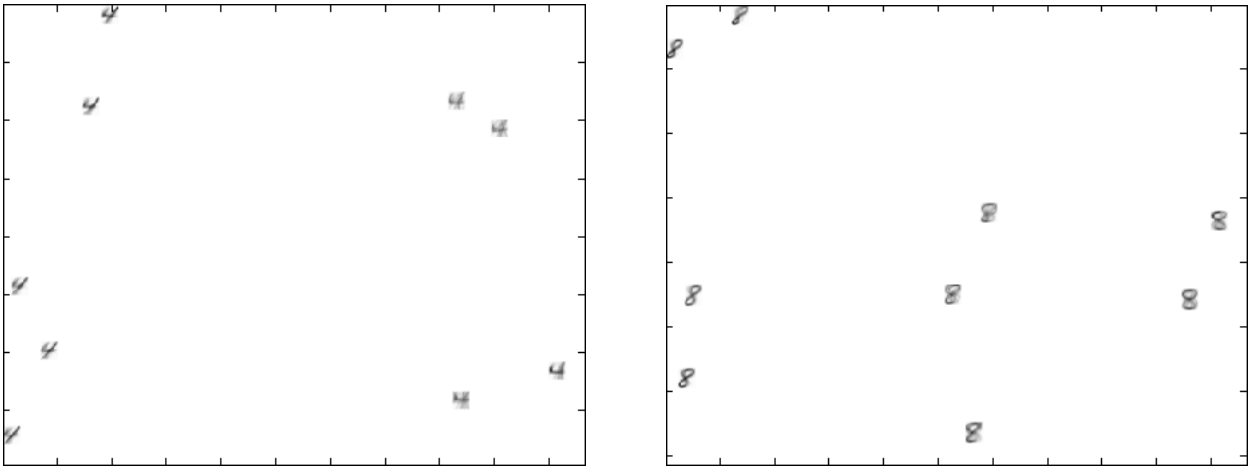


Figure 8. The topographic maps of the leaf nodes of the trees shown in Fig. 7.

VI. CONCLUSIONS

This paper proposed a growing hierarchical structure for the linear manifold topographic map network, in which the limitations of fixed topology and size of the network is removed. In the proposed model, 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 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] P. Adibi and R. Safabakhsh, "Joint entropy maximization in the kernel-based linear manifold topographic map," in: *Proceedings of the International Joint Conference on Neural Networks* (*IJCNN 07*), Orlando, Florida, pp. 1133-1138, 2007.

[2] P. Adibi and R. Safabakhsh, "Information maximization in a linear manifold topographic map," *Neural Processing Letters*, vol. 22, no. 3, pp. 155-178, 2009.

[3] P. Adibi and R. Safabakhsh, "Linear manifold topographic map formation based on an energy function with on-line adaptation rules," *Neurocomputing*, vol. 72, no. 7-9, pp. 1377-2064, 2009.

[4] P. Adibi and R. Safabakhsh, "Batch linear manifold topographic map with regional dimensionality estimation," *International Joint Conference on Neural Networks* (*IJCNN 09*), Atlanta, Georgia, pp. 63-70, 2009.

[5] P. Adibi, "Regional Linear Manifold Topographic Maps," Ph.D. dissertation, Computer Engineering Dept., Amirkabir University of Technology, Tehran, Iran, 2010.

[6] T. Hastie and W. Stuetzle, "Principal curves," *J. Amer. Statist. Assoc.*, vol. 84, pp. 502-516, 1989.

[7] R. Tibshirani, "Principal curves revisited," *Statist. Comput.*, vol. 2, pp. 183-190, 1992.

[8] J. Karhunen and J. Joutsensalo, "Representations and separation of signals using nonlinear PCA type learning," *Neural Networks*, vol. 7, no. 1, pp. 113–127, 1994.

[9] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[10] L. Saul, K. Weinberger, F. Sha, J. Ham, and D. Lee, "Spectral methods for dimensionality reduction," in *Semisupervised learning*, B. Schölkopf, O. Chapelle and A. Zien, Eds., Cambridge, MA: MIT Press, 2005.

[11] K. Diamantaras, S. Kung, Principal Component Neural Networks: Theory and Applications, New York: Wiley, 1996.

[12] P. Comon, "Independent Component Analysis - A New Concept?", *Signal Processing*, vol. 36, pp. 287-314, 1994.

[13] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin: Springer-Verlag, 2001.

[14] A. Ultsch, "Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series," in *Kohonen Maps*, E. Oja and S. Kaski, Eds., Amsterdam, The Netherlands: Elsevier, pp. 33–45, 1999.

[15] M. Su and H. Chang, "A new model of self-organizing neural networks and its application in data projection," *IEEE Trans. on Neural Networks*, vol. 12, pp. 153–158, 2001.

[16] B. Fritzke, "Unsupervised clustering with growing cell structures," in *Proc. 1991 Int'l Joint Conf. Neural Networks (IJCNN)*, Seattle, 1991.

[17] B. Fritzke, "Kohonen feature map and growing cell structures — a performance comparison," in *Proc. Adv. neural inf. Process. Syst. 5*, Denver, 1993.

[18] J. Blackmore and R. Miikkulainen, "Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map," in *Proc. 1993 IEEE Int'l Conf. Neural Networks*, 1993.

[19] B. Fritzke, "Growing cell structure — A self-organizing neural network for unsupervised and supervised learning," *Neural Networks*, vol. 7, pp. 1441–1460, 1994.

[20] B. Fritzke, "Growing grid — a self-organising network with constant neighbourhood range and adaptation strength," *Neural Process. Lett.*, vol. 2, no. 5, pp. 9–13, 1995.

[21] M. Jordan and R. Jacobs, "Hierarchical mixture of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

[22] J. Lampinen, and E. Oja, "Clustering Properties of Hierarchical Self-Organizing Maps," *Journal of Mathematical Imaging and Vision*, vol. 2, pp. 261-272, 1992.

[23] P. Koikkalainen, "Tree Structured Self-Organizing Maps," in *Kohonen Maps*, E. Oja and S. Kaski, Eds., Amsterdam: Elsevier, 1999.

[24] V. Burzevski and C. Mohan, "Hierarchical growing cell structures," in *Proc. 1996 IEEE Int'l Conf. Neural Networks*, pp. 1658–1663, 1996

[25] J. Dopazo and J. Carazo, "Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree," *J. Molecular Evolution*, vol. 44, pp. 226–233, 1997.

[26] R. Adams, K. Butchart, and N. Davey, "Hierarchical classification with a competitive evolutionary neural tree," *Neural Networks*, vol. 12, pp. 541–551, 1999.

[27] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, no. 2, pp. 126–136, 2001.

[28] M. Dittenbach, A. Rauber and, D. Merkl, "Uncovering hierarchical structure in data using the growing hierarchical self-organizing map," *Neurocomputing*, vol. 48, pp. 199-216, 2002.

[29] A. Forti, and G. Foresti, "Growing Hierarchical Tree SOM: An unsupervised neural network with dynamic topology," *Neural Networks*, vol. 19, pp. 1568-1580, 2006.

[30] M. Dittenbach, A. Rauber and, D. Merkl, "Uncovering hierarchical structure in data using the growing hierarchical self-organizing map," *Neurocomputing*, vol. 48, pp. 199-216, 2002.

[31] H. Ressom, D. Wang, and P. Natarajan, "Adaptive double self-organizing maps for clustering gene expression profiles," *Neural Networks*, vol. 16, pp. 633–640, 2003.

[32] T. Heskes, "Self-organizing maps, vector quantization, and mixture modeling," *IEEE Trans. Neural Networks*, vol. 12, no. 6, pp. 1299–1305, 2001.

[33] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in M. Jordan, Ed., *Learning in Graphical Models*, Dordrecht, The Netherlands: Kluwer, pp. 355–368, 1998.

[34] P. Demartines, and J. Hérault, "Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets," *IEEE Transactions on Neural Networks*, vol. 8, pp. 148–154, 1997.

[35] M. Tipping and C. Bishop, "Mixture of Probabilistic Principal Component Analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443-482, 1999.

[36] USPS handwritten dataset [online]. Available:
http://www.cs.toronto.edu/~roweis/data.html. Date of access: December 2010.