Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 1406–1424
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

A flexible multi-layer self-organizing map for generic processing of tree-structured data

M.K.M. Rahman, Wang Pi Yang, Tommy W.S. Chow*, Sitao Wu

Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Hong Kong

Received 27 April 2006; received in revised form 3 September 2006; accepted 8 October 2006

Abstract

A new multi-layer self-organizing map (MLSOM) is proposed for unsupervised processing tree-structured data. The MLSOM is an improved self-organizing map for handling structured data. By introducing multiple SOM layers, the MLSOM can overcome the computational speed and visualization problems of SOM for structured data (SOM-SD). Node data in different levels of a tree are processed in different layers of the MLSOM. Root nodes are dedicatedly processed on the top SOM layer enabling the MLSOM a better utilization of SOM map compared with the SOM-SD. Thus, the MLSOM exhibits better data organization, clustering, visualization, and classification results of tree-structured data. Experimental results on three different data sets demonstrate that the proposed MLSOM approach can be more efficient and effective than the SOM-SD.

Keywords: Multi-layer self-organizing map (MLSOM); Self-organizing map (SOM); Tree-structured data

1. Introduction

Neural networks have been successfully applied in a large variety of areas such as pattern recognition, etc. [1]. However, most of the traditional neural networks handle vector-type data with a fixed length. In real world, there are other complex data that cannot be represented by simple vector-type data. One type of these complex data can be usually represented by structures, in which one datum is composed of several hierarchical and related components. For instance, an image can be segmented into several regions that may be further recursively segmented into several sub-regions. As a result, the image → region → sub-region data representation is in a form of hierarchical tree structure. How to efficiently and effectively process complex structured data have become a pressing and challenging issue for researchers. Recently, there have been different neural network models proposed for tackling structured data like trees, lists, graphs etc. [2–6]. Another method proposed by Ref. [7] recursively used a multi-layer perceptron (MLP) neural network through each node of the whole data structure. The neural network is called recursive MLP. Its training steps were called back-propagation through structure [8]. It was shown that recursive MLP was able to represent and classify structured patterns [9,10]. Some recent recursive MLP neural networks are holographic reduced representations (HRR) [5], recursive auto-associative memory (RAAM) [11], labeled RAAM (LRAAM) [12–14], and recurrent and folding networks [15–17]. Despite all these work on using MLP, a long-term dependency problem arises from the recursive MLP neural networks in the case of deep graph [18,19], that is, if data graphs have deep structures, the training of recursive MLP cannot be effective.

Self-organizing map (SOM) is an unsupervised neural network approach that can be used to handle structured data. Apart from classification task, SOM provides additional analyses of data that cannot be performed by supervised models. Applications of the SOM can be found in the areas of retrieval, clustering, visualization and classification [19–23]. In Ref. [21], a two-layer SOM network has been used for processing a fixed two-level tree data, which

* Corresponding author. Tel.: +852 27887756.
E-mail address: eetchow@cityu.edu.hk (T.W.S. Chow).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.10.010

was specialized for application of image retrieval. To process generic tree data with arbitrary level, a recent neural network approach was proposed for processing structured data [24]. The network is called SOM for structured data (SOM-SD). Similar to recursive MLP neural network, a single SOM layer is recursively used for all the nodes of data graphs. However, due to architectural weakness, the SOM-SD suffers from several drawbacks which will be discussed in Section 2.2 in detail. The major problem of SOM-SD is its poor visualization map because a small fraction of neurons are used by root nodes representing tree data. This problem is also crucial for using the SOM-SD in other applications such as retrieval, clustering and classification.

In this paper, a new multi-layer self-organizing map (MLSOM) is proposed for processing tree-structured data. Basically it is an improvement of the SOM-SD. The MLSOM uses multiple SOM layers instead of a single SOM layer in the SOM-SD. The nodes at each level of trees are presented to the corresponding SOM layer for training. Such arrangement enables the MLSOM to avoid a "varying SOM input", which causes computational burden to the SOM-SD. The MLSOM is also significantly efficient compared with the SOM-SD. This is especially noticeable when node features are of different types and length at different levels of trees. In the MLSOM, root nodes, used for representing the tree data, are processed on a dedicated top SOM layer of the MLSOM. The number of all the root nodes determines the size of the top SOM layer at which neurons are used for visualization. Therefore, data visualization can be more effectively displayed in an appropriate 2-D top SOM layer, compared with a huge SOM map in the SOM-SD. Experimental results corroborate that MLSOM is much efficient compared with the SOM-SD in terms of data organization, clustering, visualization, and classification of tree-structured data.

The rest of this paper is organized as follows. Section 2 briefly describes the basic SOM and SOM-SD. Section 3 details the proposed MLSOM for processing tree-structured data. Section 4 presents experimental results and discussions. Finally the conclusion is drawn in Section 5.

2. Preliminaries

2.1. The self-organizing map

The basic SOM consists of M neurons located usually on a 2-D grid that is either hexagonal or rectangular. Each neuron i has a d-dimensional feature vector $w_i = [w_{i1}, \ldots, w_{id}]$. The iterative SOM training algorithm can be stated as follows.

Step 1: Set iteration $t = 0$.

Step 2: Randomly select a sample data vector $x(t)$ from a training set.

Step 3: Compute the distances between $x(t)$ and all feature vectors. The winning neuron, denoted by c, is the neuron with the feature vector closest to $x(t)$

$$c = \arg\min_i \|x(t) - w_i\|, \quad i \in \{1, \ldots, M\}. \tag{1}$$

Step 4: Update the winner neuron and its neighbor neurons to move its feature vector towards the data vector. The weight-updating rule in the sequential SOM algorithm can be written as

$$w_i(t+1) = \begin{cases} w_i(t) + \eta(t) h_{ic}(t)(x(t) - w_i(t)) & \forall\, i \in N_c, \\ w_i(t) & \text{otherwise}, \end{cases} \tag{2}$$

$\eta(t)$ is the learning rate which decreases monotonically with iteration t.

$$\eta(t) = \eta_0 \cdot \exp\left(-\alpha \cdot \frac{t}{\tau}\right), \tag{3}$$

where η_0 is the initial learning rate, and α is an exponential decay constant. N_c is a set of neighboring neurons of the winning neuron, and $h_{ic}(t)$ is the neighborhood kernel function that defines the closeness of a neighborhood neuron to the winning neuron c at position (x_c, y_c). The neighborhood function $h_{ic}(t)$ also decreases gradually during the learning process.

$$h_{ic}(t) = \exp\left(-\frac{[(x_i - x_c)^2 + (y_i - y_c)^2]}{2\sigma^2(t)}\right), \tag{4}$$

where $\sigma(t)$ is the width of the neighborhood function that decreases with iteration

$$\sigma(t) = \sigma_0 \cdot \exp\left(-\frac{t}{\tau} \cdot \log \sigma_0\right), \tag{5}$$

where σ_0 is the initial width, τ is a time constant which is set equal to the maximum number of iterations.

Step 5: Stop if the maximum iteration is reached. Otherwise set $t = t + 1$ and go to step 2.

Through the above iterative training procedures, different neurons become representative of different data samples. The feature vectors of neurons become topologically ordered, i.e., two neurons having similar feature vector lies in neighbor of the SOM output grid. Such organized map can be used for a number of tasks such as dimensionality reduction, data quantization, visualization, clustering and retrieval.

2.2. The SOM-SD

The SOM-SD [24] is an extension of the basic SOM such that tree-structured data can be processed through recursively use of the SOM. A tree-structured datum consists of a set of nodes organized in different levels, of which root node lies on the top and leaf nodes appear at the bottom. Except the leaf nodes, all other nodes have two types of attributes: feature vector, and "child-vector", a vector pointing to its children nodes. During the course of training, the nodes of a tree data are presented to an SOM in bottom-up

fashion, i.e., bottom-layer nodes are processed first, and root nodes are processed last because the positions of the winner neurons of the child nodes are used as the 'child-vector' for SOM input representation. Fig. 1 illustrates the encoding of a simple two-level tree datum into SOM-SD. First, the winner neurons of two leaf nodes are found on the SOM map. The positions of the winner neurons are then used together with root node's feature vector for the input representation of the root node. Any node in an intermediate level is processed in the same way.

Thus, the inputs of the shared SOM are the combination of the features of the current node and the output vectors of its children nodes. Assume that c is the maximum number of children nodes of a node in all the training data graphs. Then an input of the shared SOM can be denoted by a $n+2c$ dimensional vector $[u, p_1, \ldots, p_c]$, where u is the n dimensional feature vector of the current node, p_i is the 2-D position vector of the ith child node of the current node on the 2-D SOM output grid. If the current node has less than i children, p_i is filled with a vector $[-1, -1]$. Similarly, the weight vector of the SOM-SD is also a $n+2c$ dimensional vector.

After understanding the mapping of a graph into a 2-D SOM map, the basic two training steps of the SOM-SD can be described as follows.

Step 1 (Competition): For a node input $v(t)$ at iteration t, the nearest neuron a from v is found by slightly modifying Eq. (1) as

$$a = \arg\min_i \|\Lambda(v(t) - w_i)\|, \tag{6}$$

where w_i is the weight vector of neuron i with dimensions $n+2c$, Λ is a $(n+2c)\times(n+2c)$ diagonal matrix to balance the importance between u and $p_1, \ldots, p_c$ in the SOM input representation.

Step 2 (Cooperation): The weight vectors of neurons are updated such that all neurons are dragged toward $v(t)$ to some extent by Eq. (2).

The above process is repeated until maximum numbers of iterations are reached. The difference of the SOM-SD from the basic SOM for vector type data is that an input tree data consists of many nodes. To process a node in the SOM-SD, a special care has to be taken such that child nodes are already processed and $(p_1, \ldots, p_c)$ are available. Hence, for each data bottom layer nodes are processed first and the root nodes are processed last. Also, during each update in step 2, the child vectors $(p_1, \ldots, p_c)$ in $v(t)$ can be changed, hence they need to be recomputed in step 1.

In general, the SOM-SD has been successfully applied in symbol data such as policeman, ships and houses [24]. The SOM-SD, however, still experiences three basic problems associated with it. First, it is the computational speed. According to the example shown in Fig. 1, the values of $p_1, \ldots, p_c$ in the SOM input representation at each node are not determined before training. During the course of training, the values of $p_1, \ldots, p_c$ are changed from time to time and are needed to be recomputed. Furthermore, the number of nodes of trees is much more than that of trees. Since all the nodes are the inputs of the SOM-SD, the size of the SOM in SOM-SD should be very large. This made the computational speed of the SOM-SD very slow, i.e., it cost over seven days for the training of a SOM-SD with 29 864 nodes (or 3750 trees) [19]. Second, there is a major problem of visualizing data on a 2-D SOM output grid. It is not effective and efficient because the activated neurons by root nodes (or graphs) occupy only a small fraction of the total output grid (Fig. 2). Many non-activated neurons of a SOM are useless for visualization and represent the information of non-root nodes. Third, a problem arises with the SOM-SD when node features are different in types and length at different levels depending on application. Since all nodes are processed in a single SOM, the feature length of a neuron's weight vector needs to be $n = \max(\{n_k\}_{k=1,\ldots,L})$. In such a case, we can use a dummy feature (similar to null pointer in child vector in Fig. 1) to make node features all equal. This, however, is a waste of computation. More importantly, the different natures of feature at different levels have an adverse effect on the topological ordering neurons because the same weight vector is used to represent different attributes.

3. The MLSOM for tree-structured data

The MLSOM is proposed for handling tree-structured data. It is an improved model of the SOM-SD in a way that it delivers better visualization effect and faster computational speed. The node attributes at different levels of trees are ordinal-type node features. Unlike the SOM-SD that has only one SOM layer to process all the node data, the MLSOM uses multiple SOM layers to process tree-structured data. The comparative mapping of nodes of a tree between the SOM-SD and MLSOM are shown in Fig. 2. The number of SOM layers is equal to the maximum levels of trees. If there are maximum L levels in all the tree structures, we generate L groups of data and corresponding L SOM layers. The ith ($i=1, \ldots, L$) group of data is composed of all the ith level nodes in all the tree structures. The basic idea of MLSOM is that the SOM training is performed in a way of level by level, that is, the Lth level group of data is firstly trained by the Lth SOM layer. Similar to the SOM-SD, the SOM outputs of child nodes are used for the input representation (child vector) of a parent node. After the Lth SOM output information is filled in the (L-1)th SOM input representation, the (L-1)th SOM layer is then trained. This procedure repeats until the first level group of data is trained by the first SOM layer. Finally, the visualization of tree-structured data can be performed on the first SOM layer. The basic flow chart of training steps by MLSOM is illustrated in Fig. 3. The following describes the SOM input representation and MLSOM training in details.

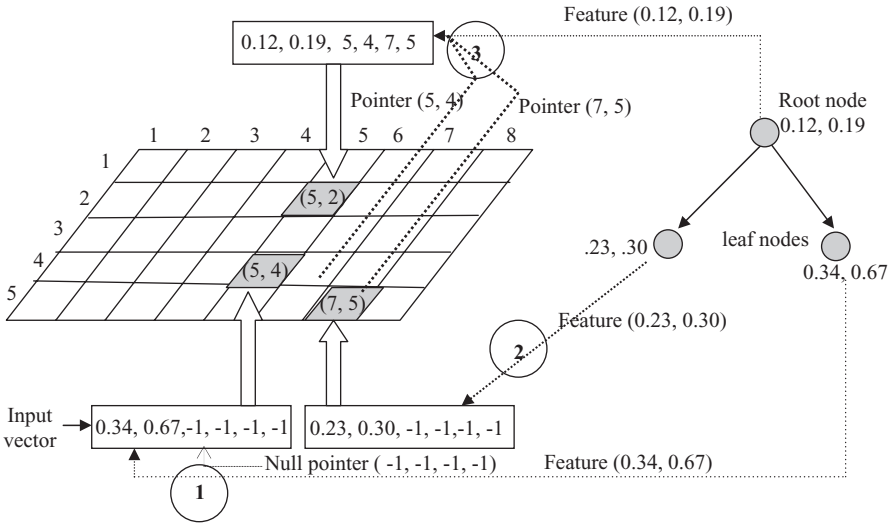

Fig. 1. Describing SOM-SD architecture how nodes are being mapped from leaf node to root node. The numbers in the big circles denote the sequence of the node processing.

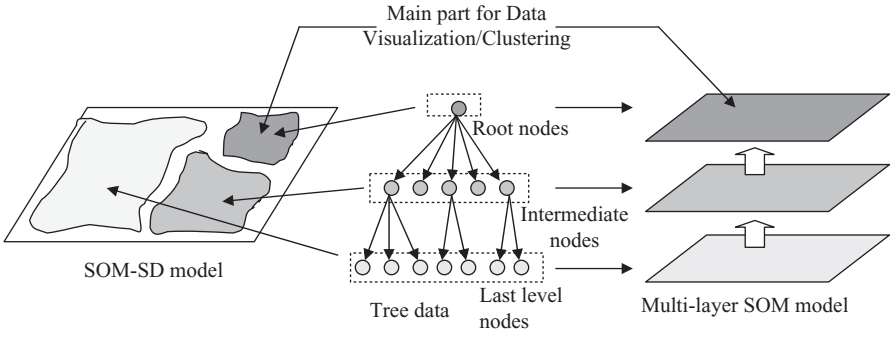

Fig. 2. The comparative data mapping and visualization in SOM-SD and multi-layer SOM model.

3.1. SOM input representation

Assume that the maximum number of children nodes of a node at the kth level in all the training data trees are c_k, the maximum levels in all the tree structures is L and the SOM output is a 2-D rectangular grid. It is noted that $c_L = 0$ as leaf nodes at the bottom layer have no children nodes. Similar to the SOM-SD, each SOM input representation at a SOM layer consists of two parts: (1) the n-dimensional feature vector u of the current node; (2) 2-D position vectors $p_1, \ldots, p_{c_k}$ of its children nodes at the SOM layer below the current SOM one. For non-leaf nodes, some p_i may be set to [0,0] since the number of children nodes of a node may be less than c_k. The lowest values of horizontal or vertical positions on the 2-D SOM output map are set to be larger than 0.

In the SOM-SD, the ordering of children nodes of a node is defined, while the ordering may not be predefined in real world applications. In the case of MLSOM, an ordering algorithm is proposed if the ordering of children is not predefined. Suppose that the nodes at the kth level need to be ordered before appending their position vectors to the SOM input representation at the (k-1)th SOM layer. After the completion of training the kth SOM layer, all the 2-D output positions of the nodes at kth level are obtained. The basic idea of our ordering algorithm is to use all these 2-D position vectors to train a 1-D SOM. The number of neurons in the 1-D SOM is set to $c_{k\text{-}1}$, i.e., the maximum number of children nodes of a node at the (k-1)th level. After the completion of training of the 1-D SOM, each training datum is assigned to a neuron index of the 1-D SOM. The neuron index is then used in the SOM input representation of $p_i (i \in \{1, \ldots, c\})$ of parent nodes at the (k-1)th SOM layer. This procedure is illustrated in Fig. 4, where the maximum number of children nodes of a node at the (k-1)th level is six. Therefore, the number of 1-D SOM neurons used for the training of output positions at the kth SOM layer is six. After the completion of training of the 1-D SOM, the 2-D weight vectors of all six neurons are marked in circles with index labels as shown in Fig. 4, and the neighboring neurons are connected with a solid line. Consider three nodes at the kth level: A, B and C. p_A, p_B and p_c are their

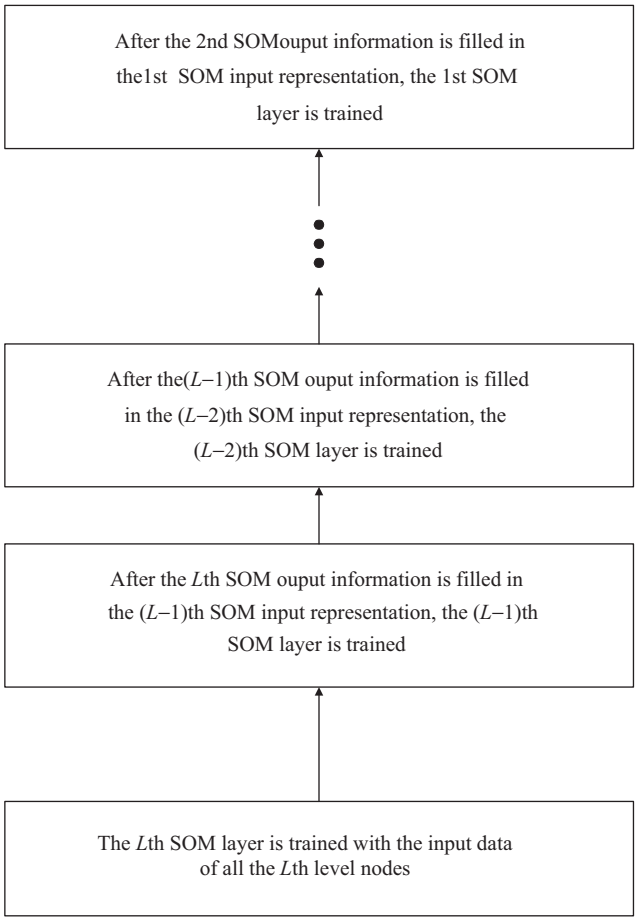

Fig. 3. The basic flow chart of training steps by MLSOM.

corresponding output positions on the kth SOM layer. The three nodes are the children nodes of a parent node at the (k-1) level. The 2-D position vectors p_A, p_B and p_c are marked in cross symbols as shown in Fig. 4. The proposed ordering algorithm just assigns p_A to its nearest neuron, i.e., neuron index 3. Then p_B is assigned to neuron index 2 and p_c to neuron index 6. Therefore, the SOM input representation $[p_1, p_2, p_3, p_4, p_5, p_6]$ of their parent node at the (k-1) level are $[\mathbf{0}, p_B, p_A, \mathbf{0}, \mathbf{0}, p_c]$, where $\mathbf{0}$ is [0,0]. This ordering can make the later similarity measurement more reasonable.

3.2. MLSOM training

Assume that each node at the kth level of trees has a n_k dimensional feature vector. The maximum of children nodes

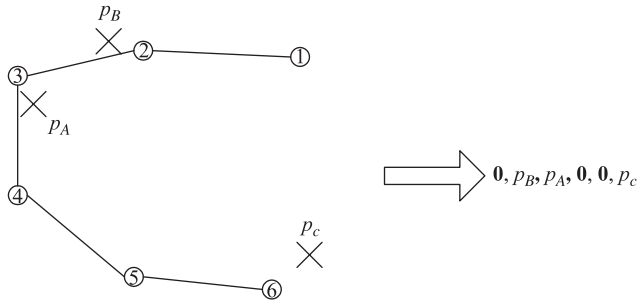

Fig. 4. The illustration of the ordering of children nodes of a node.

of a node at the kth level is c_k. The maximum levels of trees and maximum layers of the MLSOM are all L. The input data for the kth SOM layer are all $n_k + 2c_k$ dimensional vectors. There are m_k neurons at the kth SOM layer. The weights of neurons at the kth SOM layer are also $n_k + 2c_k$ dimensional vectors. Then the learning steps of MLSOM for tree-structured data are described as follows.

Step 1: Set the current level k of trees and the current layer k of a MLSOM to be L (bottom level and bottom layer).

Step 2: Normalize the 2-D positions of neurons in each SOM layer to be in the range of ((0, 1], (0, 1]).

*Step*3: Set iteration t to be 0. Collect all the nodes at the kth level nodes of trees to be the training data for the kth SOM layer. The ordered 2-D SOM output positions of children nodes of the nodes at the kth level are filled in the SOM input representation at the kth SOM layer. Therefore, the $n_k + 2c_k$ dimensional vectors are generated for the inputs of the kth SOM layer. And normalize the values in each dimension of them to be in the range of [0,1]. Randomly initialize the $n_k + 2c_k$ dimensional vectors for the m_k weights at the kth SOM layer.

Step 4: Randomly select a vector x from the $n_k + 2c_k$ dimensional vectors for the inputs of the kth SOM layer.

Step 5: Find the winner neuron a at the kth SOM layer:

$$a = \arg\max_i \|S(x, w_i^k)\|, \quad i = 1, \ldots, m_k, \tag{7}$$

where $S(x, w_i^k)$ is the similarity measurement of x and w_i^k, w_i^k is the weight vector of the ith neuron at the kth SOM layer. The similarity measurement of x and w_i^k is defined as follows:

$$S(x, w_i^k) = \frac{\lambda}{n_k} \sum_{j=1}^{n_k} \{1 - abs(x_j - w_{ij}^k)\} + \frac{1-\lambda}{\sum_{j=1}^{c_k} \delta(x_{2j+n_k-1}, x_{2j+n_k})} \sum_{j=1}^{c_k} \left\{ \delta(x_{2j+n_k-1}, x_{2j+n_k}) \times \left[1 - \sqrt{(x_{2j+n_k-1} - w_{i(2j+n_k-1)}^k)^2 + (x_{2j+n_k} - w_{i(2j+n_k)}^k)^2} \right] \right\}, \tag{8}$$

where x_j is the jth value of x, w_{ij}^k is the jth value of w_i^k, λ is weighting parameters, $\delta(x, y)$ is a function such that

$$\delta(x, y) = \begin{cases} =1 & \text{if } x \neq 0 \text{ and } y \neq 0, \\ =0 & \text{otherwise.} \end{cases}$$

The first term in Eq. (4) considers the features of the current node whilst the second one considers the compressed features of its children nodes. The weighting parameter λ determines the emphasis on global features (that appears at root nodes or higher level) or local feature (that appears lower part of tree). The choice of λ is problem-dependent. An equal weighting ($\lambda = 1$) is used in this work unless specified.

Step 6: Update the weights of neurons at the kth SOM layer by

$$w_i^k = w_i^k + \eta(t)h_{ia}(t)(x - w_i^k), \quad i = 1, \ldots, m_k, \tag{9}$$

where $\eta(t)$ is the learning rate at iteration t, $h_{ia}(t)$ is the neighborhood function around the winner node a.

Step 7: Increase the iteration by $t = t + 1$.

Step 8: If the maximum number of iterations is reached, the training at the kth SOM layer stops. Otherwise go to step 4.

Step 9: If the ordering of children nodes of a node is not predefined, the 2-D SOM output positions of nodes at the kth level are ordered by an ordering algorithm mention in Section 3.1. Otherwise, go to step 10.

Step 10: If k is equal to 1 (top level of trees or top SOM layer), the training of MLSOM stops. Otherwise, $k = k$-1, go to step 3.

From the above training steps, the second parts of the SOM input representation are fixed before training. The nodes at each level are fed into the corresponding layer for training. Therefore, the computational complexity of the MLSOM for one epoch (all tree-structured data are input to MLSOM once) is $O\left(\sum_{k=1}^{L} N_k m_k (n_k + 2c_k)\right)$, where N_k is the number of nodes at the kth level of trees, m_k is the number of neurons at the kth SOM layer, and the number of input features of a node at the kth level is n_k, and c_k is the maximum number of children nodes of all trees at the kth level. The computational complexity of the SOM-SD for one epoch is $O(Nmc(n + 2c))$ [19], where N is the number of all the nodes of trees, m is the number of neurons at the single SOM layer, n is the number of features at each node, and c is the maximum number of children nodes of all trees. Now we consider that the SOM-SD and MLSOM with same number of neurons are used for the same tree-structured data. Since $m = \sum_{i=1}^{L} m_k$, $N = \sum_{i=1}^{L} N_k$, $n \geqslant n_k$ and $c \geqslant c_k$, $\left(\sum_{k=1}^{L} N_k m_k (n_k + 2c_k)\right) < (Nmc(n + 2c))$ should be satisfied. Therefore, the computation time can be greatly reduced by MLSOM for a large training tree-structured data.

3.3. Data visualization and classification by MLSOM

The top SOM layer in the MLSOM plays an important role for visualization of the tree-structured data. The root node of a tree-structured data represents the whole data, and data visualization can be performed through their associated neurons on the top SOM layer. After the completion of training, the top SOM layer can be used for data visualization using the following procedures:

Procedures for data visualization:

1. For each tree data, find the winner neuron of the root node on the top SOM layer as follows:
   Loop from the bottom to top layer
   1. Select all the nodes at the same level as the SOM layer
   2. Find the winner neuron for each node on the current SOM layer and append the winner neuron's position vector to the input vector of its parent node.

   End
2. Save all the positions (x, y) of winner neurons from the root nodes of all tree data against the index of data.
3. Plot the data on those positions of the SOM grid. Use different symbols for different classes of data if class information is available.

By finding a winner neuron of the root node, tree-structured data are associated with that neuron on the top SOM layer. All the data can be then shown on the SOM grid using the positions of their winner neurons. Some useful information, such as clustering tendency, can be further detected from such visualization on the top SOM layer.

The visualization gives us qualitative results of the SOM. For quantitative evaluation of the SOM performance, following three numerical assessments are used.

3.3.1. Average unit disorder

Average unit disorder (AUD) [25] has been used to assess the quality of the ordering of the SOM independent of the application, and without using the class information. The AUD indicates the degree by which spatially close neurons are assigned values that are similar in the input space. The lower the value of AUD, the better the quality of self-organization. The Unit Disorder for a neuron i is defined as follows.

$$UD_i = \frac{\sum_{j=1,m} (I_{ij}/O_{ij})}{\sum_{j=1,m} (1/O_{ij})}, \tag{10}$$

where m is the number of neurons on the SOM map, input distance I_{ij} is the average absolute difference between weight vectors of two neurons i and j, and output distance O_{ij} is the Euclidean distance of those two neurons on the SOM grid. The AUD is then calculated as follows:

$$AUD = \frac{1}{m} \sum_{i=1}^{m} [UD_i]. \tag{11}$$

3.3.2. Overall Cluster Goodness

This Overall cluster goodness (OCD) is calculated with the use of class information of the data. OCD indicates how well separated are the data of different classes on the SOM

output. OCD results in a higher value when different classes form distinct clusters on the SOM visualization map. After the SOM training, the data of each class are represented by a group of representative neurons, which are shown on the visualization map of SOM. Using these representative neurons the OCD is measured in terms of two factors: (1) clusters' compactness; and (2) clusters' separation. This measurement is very similar to the work in Ref. [26]. Let $[wx_i, wy_i]$ is the normalized position vector of the winner neuron for ith data, and $\{rx_1, ry_2, \ldots rx_r, ry_r, \ldots rx_{R_c}, ry_{R_c}\}$ are a set of normalized position vectors for R_c number of representative neurons of class c. The clusters' compactness Cc is then defined as follows:

$$Cc = \frac{1}{C} \sum_{c=1,\ldots,C} \sum_{r=1,\ldots,R_c} den_{cr}, \tag{12}$$

where C is the total number of classes and density den_{cr} of a representative neuron r in class c is defined as

$$den_{cr} = \frac{1}{N_c} \sum_{i=1,\ldots,N_c} f_{ir},$$

$$f_{ir} = \begin{cases} 1 & \text{if } \sqrt{(wx_i - rx_r)^2 + (wy_i - ry_r)^2} < stdev, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

N_c is the total number of data in cth class and $stdev$ is defined as follows:

$$stdev = \sqrt{\frac{1}{C} \sum_{c=1,\ldots,C} [std(wx)^2 + std(wy)^2]},$$

where std stands for "standard deviation". The clusters' separation is simply defined by the average distance among the centers of clusters as follows:

$$Cs = \frac{2}{C(C-1)} \sum_{c=1,\ldots,(C-1)} \sum_{c_2=(c+1),\ldots,C} \times \sqrt{(mx_c - mx_{c_2})^2 + (my_c - my_{c_2})^2}, \tag{14}$$

where (mx_c, my_c) is the center of cth cluster obtained by the mean of all (wx_i, wy_i) data of that cth class. The overall cluster goodness is then formulated as

$$OCG = Cc \times Cs. \tag{15}$$

3.3.3. Average classification accuracy

The MLSOM can be further used for classification. A neuron at the top SOM layer can be labeled with a class that most associated tree-structured data belong to. In the case of tie, the class is chosen randomly. If a neuron is not assigned any datum, we just search the nearest neuron and the neuron is labeled with the class that the most structured data associated with the nearest neurons belong to. Thus, the SOM map is labeled by assigning data classes to all neurons. The labeled SOM map, called "class map", can be used for classification. When a testing tree-structured datum is fed to a MLSOM, a best-matching neuron at the top SOM layer can be found according to Eq. (7). As the best-matching neuron is already labeled with a class, the testing datum is classified to that class. Average classification accuracy (ACA) is then found by

$$ACA = \frac{\text{Number of data correctly classified}}{\text{Number of data tested}} \times 100. \tag{16}$$

4. Experimental results

In this section, three data sets were used to demonstrate the effectiveness and efficiency of the proposed MLSOM for handling tree-structured data. The first data set consists of synthetic "human" symbols. The data set can be divided into four well-separated categories and further 12 classes that can be used to demonstrate the better visualization effect of the MLSOM, compared with the SOM-SD. The second data set consists of documents and the third data set consists of real flower images. These two real data sets were used to demonstrate the better visualization and classification results of the MLSOM compared with the SOM-SD. All the presented simulations were performed using MATLAB 6.5 on a PC with P-4 1.3 GHz and 512 M memory. For the MLSOM and SOM-SD, the initial learning rate was set to 0.3 for the training of a SOM layer. The initial radius of the neighborhood function was set to half-length of the square grid at a SOM layer. The number of total training epochs (one epoch means that all the input data are fed into neural networks once) for a SOM layer was set to 10. Thus, the total number of iterations is equal to the number of nodes multiplied by the number of epochs. The values of the above parameters were observed to be a good choice. This will be clarified using the experimental results in Section 4.3.

4.1. Synthetic "human" symbol data set

The first data set consists of 432 synthetic "human" symbols. There are four categories in the symbol data. Each category of the "human" symbols has 108 data. The example from each category can be seen in Fig. 5. Furthermore, each category can be equally divided into three classes according to the positions of "human" arms. Thus, there are totally 12 classes and each class contains 36 data. The difference among the 12 classes can be seen in Table 1.

A tree structure is extracted from each symbol in the data set. The maximum number of levels in a tree is four as shown in Fig. 5. Therefore, the MLSOM has four corresponding SOM layers. The root node of a tree represents the whole "human" symbol. The second level nodes of a tree represent the local regions of a "human" symbol such as head, body, lower part, etc. The third level nodes of a tree represent the local regions of the second level nodes. And the fourth level nodes of a tree represent the local regions of the third level nodes. For example, in category 1 as shown in Fig. 5(a), the second level node "head" is

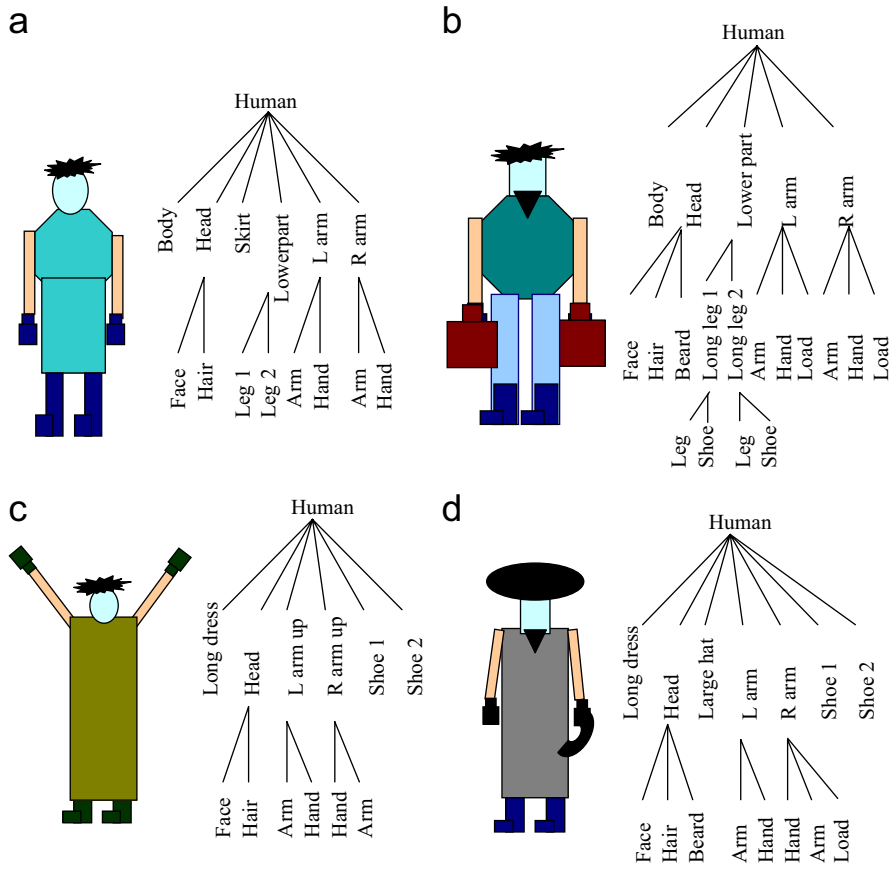

Fig. 5. Examples from four categories of the synthetic "human" symbol data: (a) Category 1; (b) Category 2; (c) Category 3; and (d) Category 4.

Table 1
Description of four categories and 12 classes of the synthetic "human" symbol data set

	The second level nodes that differentiate among categories	Class	Features of nodes (left-arm and right-arm) that differentiate among classes
Category 1	Body, head, long-skirt, lower-part, left-arm, right-arm	1	Arms are placed naturally
		2	Arms are raised above
		3	Arms are holding load/objects
Category 2	Body, head, lower-part, left-arm, right-arm	4	Arms are placed naturally
		5	Arms are raised above
		6	Arms are holding load/objects
Category 3	Long-dress, head, left-arm, right-arm, shoe, shoe	7	Arms are placed naturally
		8	Arms are raised above
		9	Arms are holding load/objects
Category 4	Long-dress, face, large-hat, left-arm, right-arm, shoe, shoe	10	Arms are placed naturally
		11	Arms are raised above
		12	Arms are holding load/objects

composed of "hair" and "face" nodes at the third level. Each node of a tree is denoted by four features: (1) size; (2) shape; (3) the horizontal coordinate; and (4) the vertical coordinate. For a non-leaf node, its children nodes have been already ordered by us. For an example shown in Fig. 5(a), a root node in categories 1 has six children nodes. The second part of the SOM input at the root node is represented by $[p_{Body}, p_{Head}, p_{Skirt}, p_{Lower\ part}, p_{L\ arm}, p_{R\ arm}]$. As shown in Fig. 5(a), the second part of the SOM input at the "head" node is represented by $[p_{face}, p_{hair}, \mathbf{0}, \ldots, \mathbf{0}]$. As shown in Fig. 5(b), the second part of the SOM input at the "head" node is represented by $[p_{face}, p_{hair}, p_{Beard}, \mathbf{0}, \ldots, \mathbf{0}]$. Therefore, the ordering procedure mentioned in Section 3.1 is not used in the synthetic "human" symbol data. The node

Table 2
Description of four categories of the first document data set

	Some discriminative keywords of different categories (a keyword not necessarily appeared in all documents of the related category)
Category 1	Health, recipe, instruction, meat, cook, food,
Category 2	Hair, skin, cosmetic, beauty, fashion, care, model
Category 3	Family, guard, dog, pet, home, breed, protect
Category 4	Facial, feature, recognition, image, face, search, database

distribution and the maximum number of child node at different level are detailed in Table 2.

According to the numbers of nodes at different levels of trees, the sizes of the corresponding the first, second, third and fourth SOM layers in MLSOM were firstly set to 5×5, 10×10, 12×12 and 5×5, respectively. The SOM-SD was also used for comparison with a 17×17 SOM due to the large size of all nodes. The visualization of the MLSOM and SOM-SD is shown in Fig. 6(a) and (d), respectively, where different categories are represented by different symbols and different classes in a category are represented by the same symbol but different symbol sizes. In the case of MLSOM, the four categories are well separated from other categories. The classes in categories 1 and 2 can be separated from other classes. But some classes (e.g., classes 7, 9–12) in categories 3 and 4 are overlapped by the other classes. Ten activated neurons at the top SOM layer are fully distributed on the 2-D grid. In the SOM-SD, only five neurons are activated and concentrated at the top-right corner of the 2-D grid. Like the MLSOM, the four categories can be well separated. But only class 11 is separated from other classes. The other classes are totally overlapped. The visualization of the SOM-SD is not good since the visualization area for data (top-right corner) occupies a small part of the whole SOM output grid. There are still a lot of neurons not utilized for the visualization of data.

If the SOM sizes at the first, second, third and fourth SOM layer in the MLSOM are increased to 10×10, 20×20, 24×24, and 10×10, respectively, the visualization of the MLSOM becomes more precise and clear as shown in Fig. 6(b). Nineteen neurons at the top SOM layer are activated for visualization. Each category is well separated from other categories. And each class in a category can be well separated from any other classes. For the SOM-SD, if the single SOM size is increased to 35×35, the visualization result is shown in Fig. 6(e). Still there are only five neurons activated for visualization. The four categories are still separated from other categories. But only class 5 is separated from other classes. The other classes are totally overlapped. Similarly, the visualization of SOM-SD is not good due to the low utilization of neurons for visualization.

If the SOM sizes at the first, second, third and fourth SOM layer in the MLSOM are increased to 15×15, 30×30, 36×36, and 15×15, respectively, the visualization becomes much more precise as shown in Fig. 6(c) than the case shown in Fig. 6(b). There are totally 26 neurons activated for visualization. Different categories and classes are well separated in the visualization, and each class can be represented by one or more neurons. For the SOM-SD with a 50×50 SOM size, the visualization is shown in Fig. 6 (f). The precision is improved a little. The four categories are still well-separated. Only classes 10 and 12 cannot be separated from each other. Other classes are well-separated. For a very large SOM grid with 2500 neurons, only 16 neurons are utilized for visualization. Therefore, the visualization of the SOM-SD is still not satisfactory since the 12 classes cannot be well separated.

From the above experiments, the size of the top SOM size can be easily adjusted according to users' precision requirement of visualization for the MLSOM. However, for the SOM-SD, even if the size of the single SOM is greatly increased, it is difficult to obtain a more precise visualization because all the nodes are used for the single SOM layer and the number of root nodes is a very small fraction of the total nodes.

From the visualization using the MLSOM, we can learn some clustering tendency of 12 classes. For example, in Fig. 6(c), the 12 classes are well separated from each other for MLSOM. From the visualization, each class can form its individual cluster. But for SOM-SD, only the clustering tendency of four categories can be detected (e.g., Fig. 6(f)) even if a very large map size (i.e., 50×50) is used.

4.2. The document data set

At first we considered a document data set of four classes for a detailed experimental analysis. Later, we increased the data set to 40 classes to evaluate classification accuracy on larger data set. The first document data set consists of 400 text documents in 'html' format downloaded from the web. The document data are divided into four categories and each category contains 100 documents. Some discriminative features of these four categories are listed in Table 2. A three-level tree-structured data were extracted in a manner that the root node represents the whole document, the second level nodes represent different pages, and the third level nodes represent the paragraphs of the pages. Fig. 7 illustrates tree-structured feature representation of a document. Node features at different level are represented by histogram of automatically generated keywords. Two documents having similar word-histograms at root nodes can be completely different in terms of semantics/context. It is because of different orientations of the same set of words throughout the document, which is reflected by the discriminative lower parts of the tree data (second/third level modes). Thus, tree-structured features can help a better analysis of the documents. To extract the document features, text are separated from html tags, and stop words (set of common words like "the", "is" etc.) are removed. Porter stemming algorithm [27] is used to extract stem of each word, and stems are used

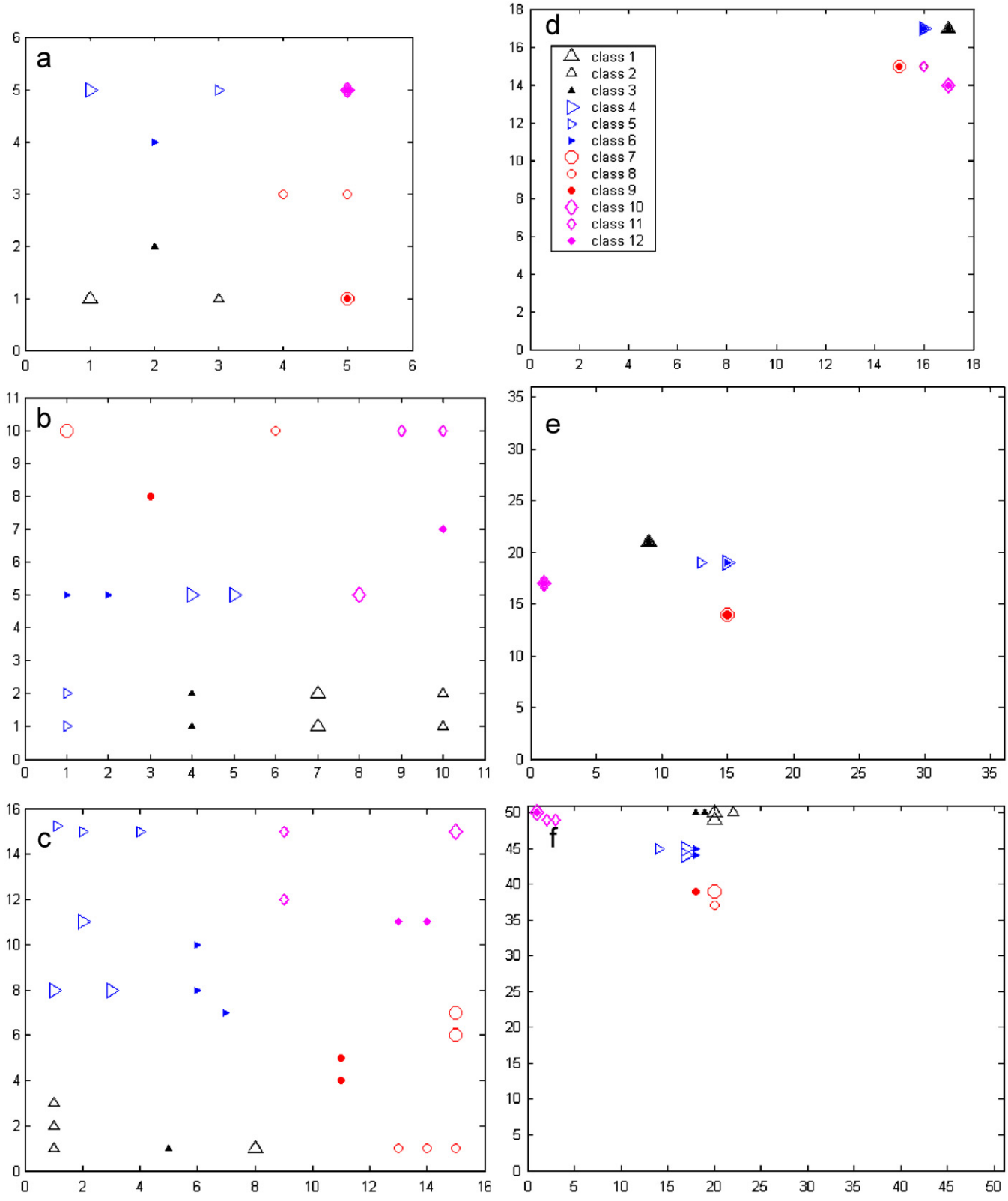

Fig. 6. Visualization of the synthetic "human" symbol data. (a) MLSOM (5 × 5 – 12 × 12 – 10 × 10 – 5 × 5); (b) MLSOM 10 × 10 – 24 × 24 – 20 × 20 – 10 × 10); (c) MLSOM (15 × 15 – 36 × 36 – 30 × 30 – 15 × 15); (d) SOM-SD (17 × 17); (e) SOM-SD (35 × 35); and (f) SOM-SD (50 × 50).

for feature extraction instead of original words. Then a word list is build up for the whole data set which stores term frequency f_T (the frequency of a word in all documents) and document frequency f_d (the number of documents a word appeared). A weight of each word is then calculated from inverse document frequency *idf* and f_T, which is very similar to the term weighting in vector space model [28].

$$W_{term} = idf \times \sqrt{f_T}, \tag{17}$$

where $idf = 1/e^{((8N_c/N)(N/f_d-2))}$, N_c is the number of data in a class, and N is the total number of data in whole data set. Words having higher weights are selected first as important words and different numbers of keywords are selected for nodes at different level as shown in Fig. 7. Using the html tags, the documents are partitioned into pages, and the pages are partitioned into paragraphs. As a paragraph contains fewer words, more keywords are used to describe the third level nodes. Histogram of a document/page/paragraph is computed as follows:

$$H_T = [h_1 h_2 h_3 \ldots\ldots h_T], \quad h_t = \frac{n_t}{\sum_{t=1,\ldots,T} n_t}, \tag{18}$$

where T is the total number of keywords used, and n_t is the frequency of tth keyword.

Node distribution of document data set is listed in Table 3. For a non-leaf node, the children nodes have not been already ordered. Therefore, the ordering procedure mentioned in Section 3.1 was used for children node at the first and second SOM layer. In the MLSOM, the numbers of features of nodes at different levels of trees, i.e., $n_k(k = 1, 2, 3)$, can be different. But in the SOM-SD, the numbers of features of nodes at different levels of trees should be the same. In order to make the SOM-SD suitable for processing the structured data extracted from documents, the number of node features in SOM-SD is chosen to be the maximum number (i.e., 100) of features of nodes at all the levels of trees. If the number of features of a node is less than 100, we just append zeros to the end of its features. Thus, the number of features of nodes at all the levels can be the same value of 100 for the SOM-SD. In addition, children nodes in the SOM-SD are ordered according to their output positions on the single SOM layer from top to bottom and from left to right.

For training of the MLSOM and SOM-SD 200 documents are used, each category containing 50 documents. The

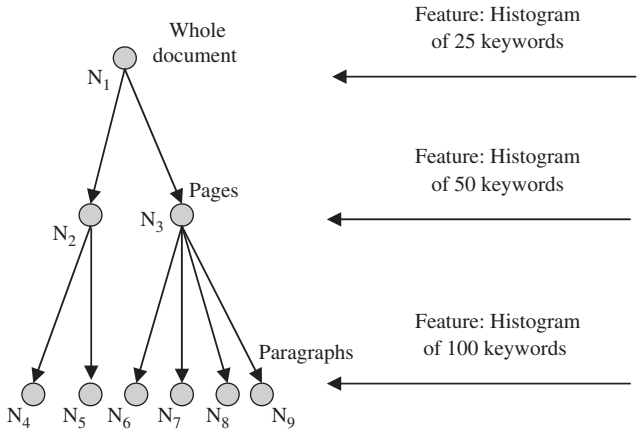

Fig. 7. Illustration for the tree-structured feature representation of a document.

Table 3
Distribution of nodes in three data sets that are used in the experiment

Level	1	2	3	4	All levels
1. *The synthetic "human" symbol data set* (*totally* 432 *tree-structured data*)					
Maximum number of children nodes	7	3	2	0	7
Total number of nodes in training set	432	2592	3204	432	6660
2*a*. *The document data set* (4 *classes*, *totally* 400 *tree-structured data*)					
Maximum number of children nodes	18	57	0	—	57
Total number of nodes in training set	200	452	2989	—	3641
Total number of nodes in testing set	200	433	2824	—	3457
2*b*. *The document data set* (40 *classes*, *totally* 4000 *tree-structured data*)					
Maximum number of children nodes	69	127	0	—	127
Total number of nodes in training set	2000	3297	15 937	—	21 234
Total number of nodes in testing set	2000	3308	16 084	—	21 392
3*a*. *The flower image data set* (12 *classes*, *totally* 480 *tree-structured data*)					
Maximum number of children nodes	4	8	0	—	8
Total number of nodes in training set	240	555	1425	—	2220
Total number of nodes in testing set	240	549	1342	—	2131
3*b*. *The flower image data set* (36 *classes*, *totally* 1440 *tree-structured data*)					
Maximum number of children nodes	4	8	0	—	8
Total number of nodes in training set	720	1592	4181	—	6493
Total number of nodes in testing set	720	1573	4150	—	6443

rest 200 documents are used as a testing set in the evaluation of classification accuracy. After the training of SOM, the visualization of the training data set using a MLSOM (20×20–24×24–28×28) and a SOM-SD (40×40) are shown in Fig. 8(a) and (b), respectively. Once again, with a smaller grid of 20×20, MLSOM can deliver much better visualization for the four categories of documents. Different categories formed almost separate clusters on the SOM from which 'distinct class boundaries' can be drawn for analysis of new data. On the other hand, even with a bigger SOM grid of 40×40 fewer neurons are activated on the SOM-SD leading to a poor visualization. Fig. 8(c) and (d) show the class maps for the MLSOM and SOM-SD, respectively. Class map shows labels of neurons as discussed in Section 3.3.3. Ignoring few isolated neurons, the MLSOM formed four distinct class regions. On the other hand, class regions are mostly subdivided into more than one part on the SOM-SD. Visualization of testing data set using the MLSOM and SOM-SD are shown in Fig. 8(c) and (d), respectively. A comparative inspection on Fig. 8 explains that data analysis is much effective and easier on the MLSOM. With distinct class regions it is easier to comprehend the nature of new data from their mapping on the SOM. Using the class maps the classification results on the training and the testing sets are summarized in Table 4. With a much less computation, the MLSOM achieved better classification accuracy than the SOM-SD. Obviously, the higher accuracy is related to the better class map of the MLSOM.

We have also summarized some quantitative performance measure against the SOM size, which is the most crucial among SOM parameters. Fig. 9 summarizes the AUD, OCG and ACA performance of the MLSOM and SOM-SD for different SOM size. A lower AUD value reflects a better relationship between a low-dimensional SOM output and high dimensional input data. Thus, a better understanding and analysis of data is possible with a lower AUD value. Fig 9(a) shows that the MLSOM can deliver consistently better AUD value at different SOM sizes compared with the SOM-SD. The higher AUD value of SOM-SD can be explained by the mixing up of different levels' nodes in the same SOM layer. The OCG value reflects the clustering tendency of the data on the SOM, which in turn indicates the quality of visualization. Fig. 9(b) shows the OCG performance on document data set against different SOM sizes. Finally, ACA is plotted against the SOM sizes in Fig. 9(c) and (d) for the training and the testing sets, respectively. It can be noticed that the performance of SOM-SD rapidly decreases with the decrease in SOM size. In contrast, the MLSOM maintains a stable and superior performance at different SOM sizes.

To evaluate classification performance on a larger data set, we increased the document classes from four classes to 40 classes, each class containing 100 documents. This data set is made much harder for classification task as the classes share many discriminative keywords among themselves, which is different from the case of the previous database described in Table 2. Tree structured features were made up using word histogram of 1000 keywords in each level. Because the feature dimension was too big, we used standard PCA (principal components analysis) to compress the feature of first, second and third level nodes into a reduced dimension of 100, 150 and 200, respectively. Using this data set, the classification performance of the MLSOM and SOM are summarized in Table 4. The SOM-SD performed rather poorly by delivering a classification accuracy of 19% and 15% on the training and the testing sets, respectively. On the other hand, the MLSOM delivered a much respectable classification accuracy of 70% and 52% on the training and testing sets, respectively. Also, it should be noted that the training time of the SOM-SD is 33 times higher than that of the MLSOM. The performance of the SOM-SD can be further increased by increasing the SOM size but at a cost of extreme computational time.

4.3. Flower image data set

The first flower data set consists of 480 real flower images. There are totally 12 different species of flowers. The image samples from 12 different flower species are shown in Fig. 10. Each species has 40 flower images. All the flower images were divided into training and testing sets. The total number of flower images used for training is 240 and that for testing is 240. A flower image is represented by a three-level tree, which is generated by image segmentation. The root node of a tree represents the whole image. The second level nodes of a tree represent local regions such as background and flowers. The third level nodes of a tree represent more specific local regions such as each individual flower.

For the experiment on this data set, structured data were extracted from the flower in a tree representation. Tree-based representation is shown to be useful in image analysis [29,30] because it includes the spatial relationship among various objects in an image. Fig. 11 demonstrates how a flower image is represented by a three-level tree through a two-level image partitioning. In the first level partitioning shown in Fig. 11(b), three HSV channels of the image are used to cluster image pixels in the color domain. Fig. 11(c) demonstrates the second-level partitioning in the image, where non-connected regions with similar color are obtained. The extracted features from the image or regions serve as the features of nodes of a tree. Color histogram (16 bins for each of HSV channels) is used as the features of root nodes. The color moments, i.e., 3 averages and 3 standard deviations of HSV channels, are used for region features of the second level nodes. In order to describe the shape features used for non-root nodes, normalized inertia of orders 1–3 are used [31]. Moments of Haar wavelet coefficients in three different frequency bands are used as texture features (3 averages and 3 standard deviations) of the third level nodes. Details of these feature extractions can be found in [31,32]. Therefore, the root nodes are denoted by $48 (= 16 \times 3)$ features of color histograms. The second level nodes of a tree are denoted by 10 features: 6 features

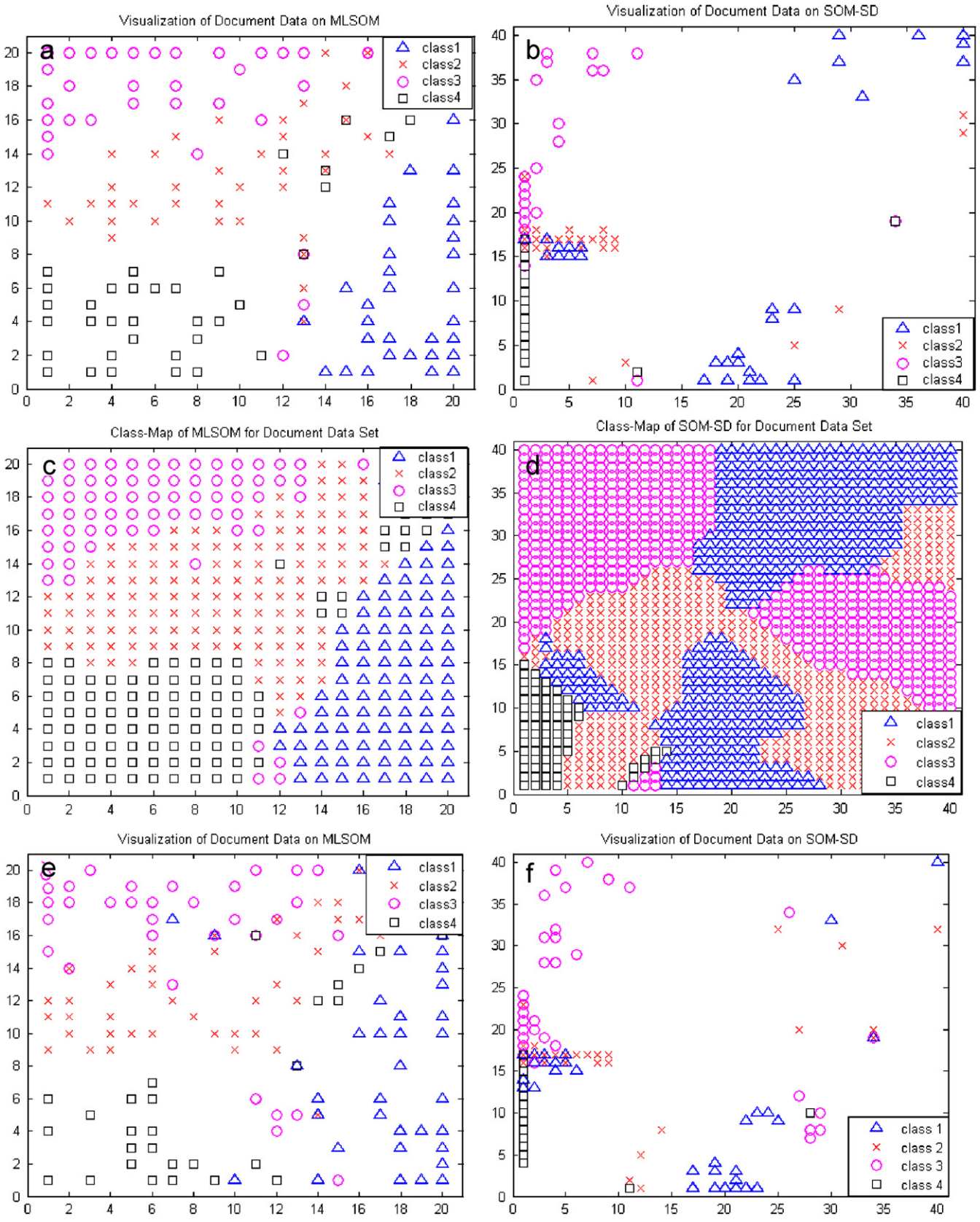

Fig. 8. Visualization of the four categories of document from training set (a) MLSOM ($20 \times 20 - 24 \times 24 - 28 \times 28$) and (b) SOM-SD ($40 \times 40$); and class map of document database formed on (c) MLSOM and (d) SOM-SD. Visualization of testing set (e) MLSOM and (f) SOM-SD.

Table 4
Training time and classification performance using the flower image data set and the document data set

Methods	SOM size	Training time (in minutes)	Classification accuracy	
			On training data	On testing data
1. *The document data set* (4 *classes*)				
MLSOM	20 × 20–24 × 24–28 × 28	16.8	94	81
	30 × 30–36 × 36–42 × 42	69.5	96	84
SOM-SD	30 × 30	258.2	82	70
	40 × 40	611.7	88	76
2. *The document data set* (40 *classes*)				
MLSOM	30 × 30–36 × 36–42 × 42	291.9	70	52
SOM-SD	40 × 40	9833.3	19	15
3. *The flower image data set* (12 *classes*)				
MLSOM	20 × 20–24 × 24–28 × 28	6.7	95	86
	40 × 40–48 × 48–56 × 56	14.5	100	93
SOM-SD	30 × 30	30.1	87	71
	80 × 80	184.0	90	75
4. *The flower image data set* (36 *classes*)				
MLSOM	20 × 20–24 × 24–28 × 28	8.2	90	81
SOM-SD	30 × 30	153.9	69	58

of color moments, 1 feature of size, 3 features of shapes. The third level nodes of a tree are denoted by 10 features: 6 features of textures, 1 features of size, 3 features of shapes. The ordering procedure mentioned in Section 3.1 was used for a non-leaf node at the first and second SOM layer, where the children nodes have not been already ordered. Table 3 summarizes the node distribution at different levels as well as the maximum number of child node. Similar to the document data set, the number of node features in SOM-SD is chosen to be the maximum number (i.e., 48) of features of nodes at all the levels of trees.

According to the numbers of nodes at different levels of trees, the sizes of the corresponding the first, second and third SOM layers in the MLSOM were empirically set to 10 × 10, 12 × 12 and 14 × 14, respectively. SOM-SD was used for comparison with a 20 × 20 SOM due to the large number of all the nodes. After the completion of training, the visualization of 12 flower species on the SOM is shown in Fig. 12(a) and (d) for the MLSOM and SOM-SD, respectively. As seen from Fig. 12(a), almost all the neurons at the top SOM layer were used for visualization. There is some overlapping among all the 12 flower species. As seen from Fig. 12(d), in SOM-SD, the 12 flower species are mapped into a small area of the single SOM map. All species are heavily overlapped.

If the SOM map sizes at the first, second and third SOM layer of the MLSOM are increased to 20 × 20, 24 × 24 and 28 × 28, respectively, the visualization becomes more precise and clear as shown in Fig. 12(b). Some species (i.e., species 2 and 10) are separated from other species. There is still some overlapping among other species. For the SOM-SD, if the single SOM size is increased to 40 × 40, the visualization result is shown in Fig. 12(e). Still, there is a small area of the single SOM layer for visualization. The flower species are also heavily overlapped.

If the SOM sizes at the first, second and third SOM layer of the MLSOM are increased to 40 × 40, 48 × 48 and 56 × 56, respectively, the visualization becomes much more precise as shown in Fig. 12(c) than the case shown in Fig. 12(b). All the 12 species are well separated from other species. Species 6, 7 and 11 form distinct clusters. For the SOM-SD, if the single SOM size is increased to a very large size 80 × 80, the visualization result is shown in Fig. 12(f). Some neurons with species 10 form a cluster. There is some overlapping among all 12 species due to the small area for visualization on the single SOM layer. Compared with data set 1, the worse visualization of the SOM-SD for the flower data set can be explained by the third problem mentioned at the end of Section 2.2, i.e., node features are different in types and length at different levels. In addition, the ordering of child nodes in the MLSOM is more meaningful compared with the SOM-SD, because the SOM outputs of child nodes are fixed during the training process.

We also performed classification on the 12 flower species for the MLSOM and SOM-SD. The classification results for different SOM sizes are summarized in Table 4. The results show that with much lesser computational costs, the MLSOM delivers better classification performance on both the training and testing data sets than the SOM-SD. The better classification accuracy can be easily explained with the superior visualization of tree-structured data on the MLSOM map. In Table 4, we have also included flower classification results using a larger data set of 36 flower species. Similar to previous data set, each species contains 40 image data, and the data set equally divided into the training and the testing set. The node distribution of this data set is shown in

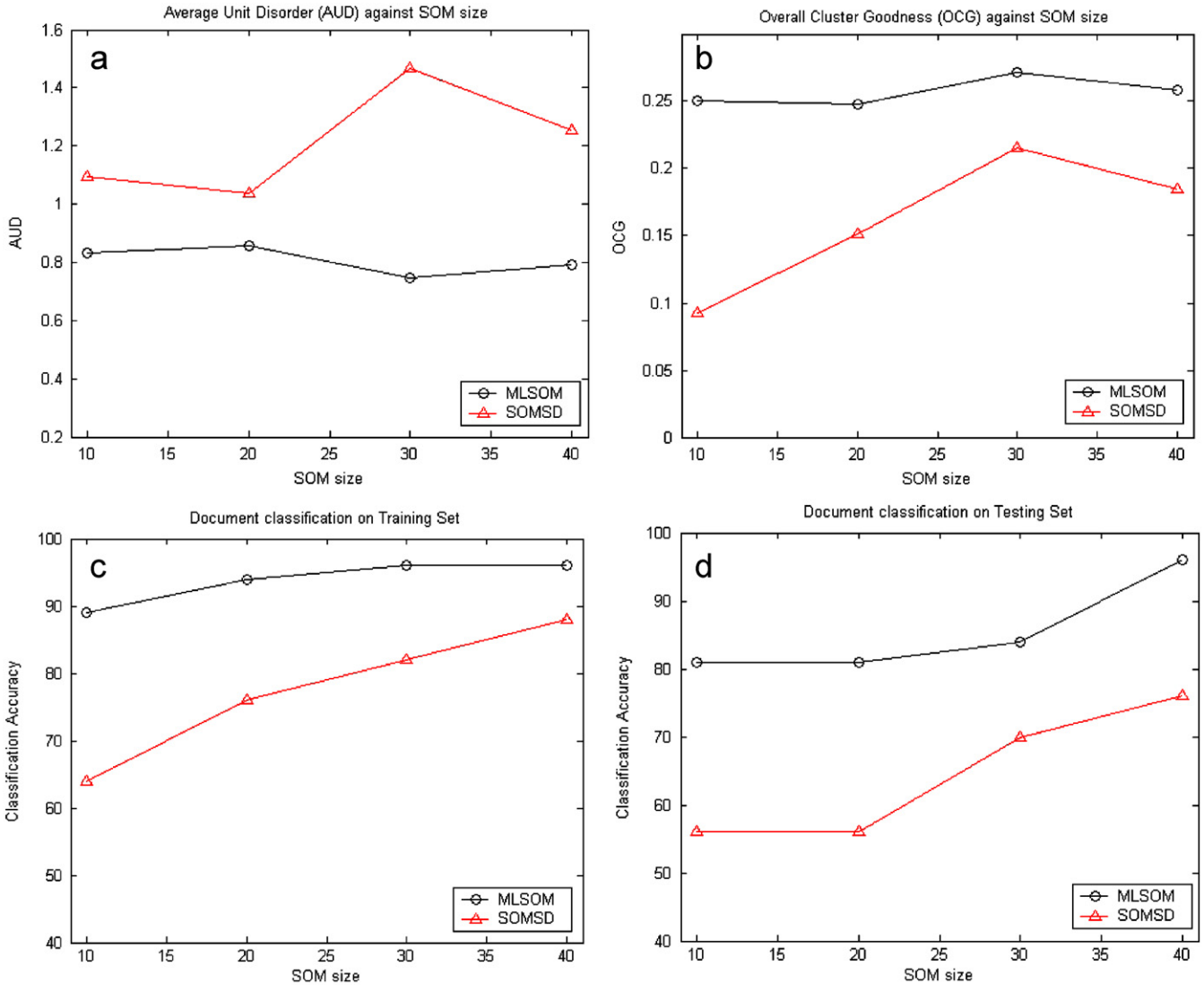

Fig. 9. Quantitative performance comparisons against SOM size (top layer for MLSOM) using the document database. Average classification accuracy against SOM size (length of square SOM grid): (c) training set and (d) testing set. (a) average unit disorder against SOM size. (b) overall cluster.

Table 3. Once again, the MLSOM delivers superior results in terms of classification accuracy and computational time.

We also used the flower data set of 12 species to analyze the comparative results between the MLSOM and SOM-SD under different SOM settings such as different initializations, learning rate and number of epochs. The size used for the MLSOM and the SOM-SD were 20×20–24×24–28×28 and 30×30, respectively. The initial learning rate and the number of training epochs are set to 0.3 and 10, respectively as default setting. Fig 13(a) shows the classification accuracy using five random initializations of SOM weight vectors before training. Results of the MLSOM and SOM-SD are more or less similar under five initializations indicating a stable training of both models. Fig. 13(b) shows the results under five different initial learning rates. An initial learning rate of 0.3 or higher seems to be a good choice for both networks. Finally, results under different number of training epochs are shown in Fig. 13(c) that indicate 10 epochs are fairly good enough. In all the cases, the MLSOM consistently delivers better performance than the SOM-SD in both training and testing set.

For comparative neural network sizes, the computational training time for the MLSOM is compared with that of the SOM-SD. These are listed in Table 4. Clearly the MLSOM is also significantly much faster than the SOM-SD. The most important factor contributes to the relatively fast training process is that 48 features were used for all the nodes in the SOM-SD whilst only 10 features were used for non-root nodes in the MLSOM.

The computational time difference between two methods in Table 4 can be easily comprehended by comparing the computational complexity of MLSOM $O\left(\sum_{k=1}^{L} N_k m_k (n_k+2c_k)\right)$, and SOM-SD $O(Nmc(n+2c))$ as discussed in

Fig. 10. Samples from 12 different species of flower images.

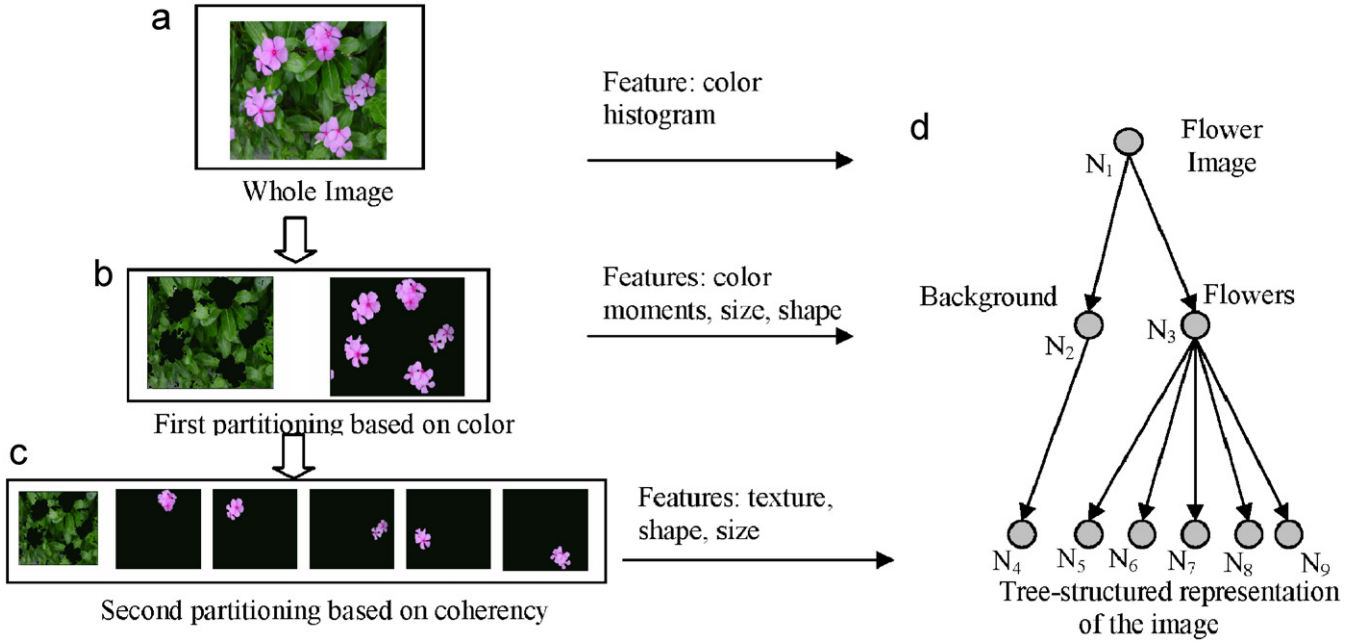

Fig. 11. Demonstration of a tree-structure from a flower image. (a)–(c) partitioning flower images; and (d) tree-structured image representation.

Section 3.2. In an extreme case, we can consider a SOM size for each layer of MLSOM equal to the size of SOM-SD $(m = m_k)_{k=1,\ldots,L}$, and n_k and c_k are same for all levels $(c = c_k,\ n = n_k)_{k=1,\ldots,L}$. The computational complexity of the MLSOM becomes $O\left(\sum_{k=1}^{L} N_k m(n+2c)\right) \Rightarrow O\left(m(n+2c)\sum_{k=1}^{L} N_k\right) \Rightarrow O(Nm(n+2c)\right)$. This means that with L times more neurons the computation complexity of MLSOM is still less than that of the SOM-SD $O(Nmc(n+2c))$. Comparative computational demand of the SOM-SD further increases with the difference between n and n_k, and with the difference between c and c_k at different levels. In other words, when the feature length (n_k) and the number of child nodes (c_k) are not the same at different levels, the SOM-SD needs to pay some extra cost.

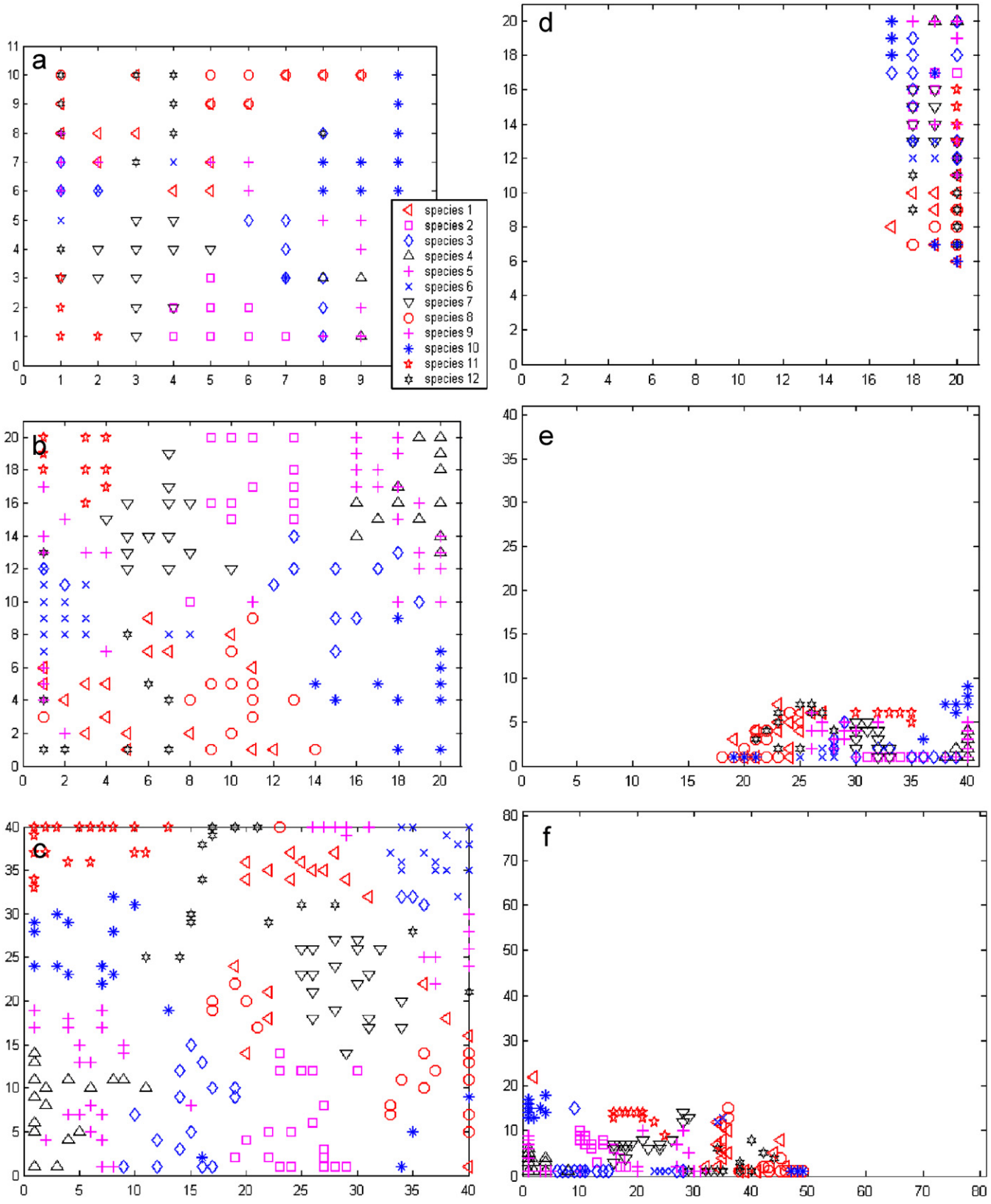

Fig. 12. Visualization of 12 different flower species by (a) MLSOM (10 × 10 – 12 × 12 – 14 × 14); (b) MLSOM (20 × 20 – 24 × 24 – 28 × 28); (c) MLSOM (40 × 40 – 48 × 48 – 56 × 56); (d) SOM-SD (20 × 20); (e) SOM-SD (40 × 40); and (f) SOM-SD (80 × 80).

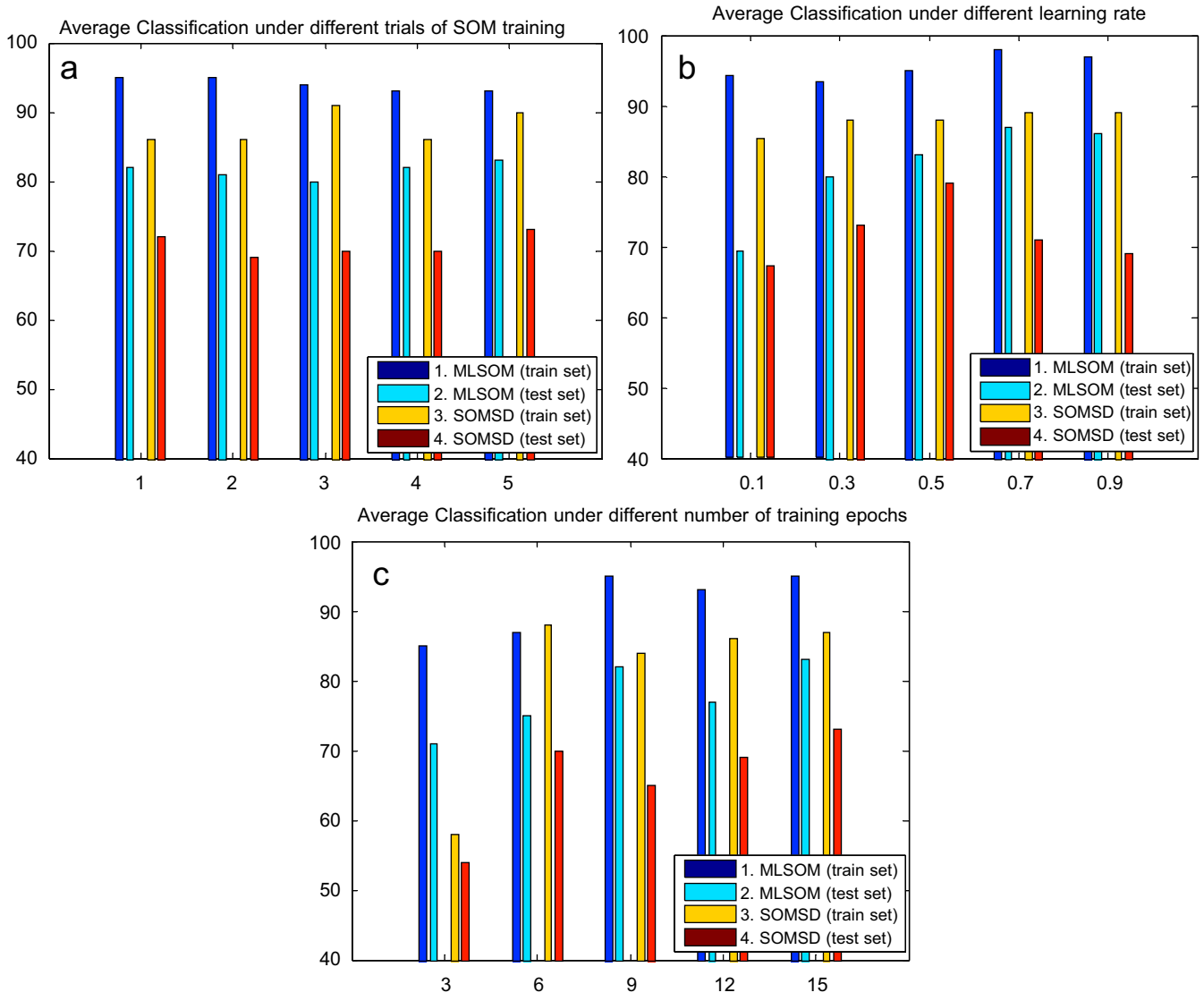

Fig. 13. Comparative classification accuracy between MLSOM and SOM-SD under different settings: (a) SOM initializations; (b) initial learning rate; and (c) number of training epochs.

5. Conclusion

A new SOM structure is proposed for processing tree-structured data. Unlike the SOM-SD that processes all the nodes of tree in a single SOM layer, the proposed MLSOM uses multiple SOM layers in a way that it processes all the nodes at a certain level in the corresponding SOM layer. Thus, the output of the MLSOM is not adversely affected like the SOM-SD when node features at different level of the tree have different nature and length. Another architectural advantage of the MLSOM over SOM-SD is that it does not require pre-processed ordering of the child nodes in the tree-structured data. Unlike the SOM-SD, the input of the MLSOM does not change with time during training process, and the number neurons used for data visualization are not adversely affected by total number of nodes. All these features enable the MLSOM to provide a much efficient training and organization of tree-structured data. Our studied results indicate that the MLSOM delivers significantly better visualization effects compared with the SOM-SD. It is also worth noting that the MLSOM is much computationally efficient than the SOM-SD. The classification performance delivered from the MLSOM is superior to that of the SOM-SD. Experimental results corroborate that the MLSOM is a very efficient approach for processing tree-structured data in terms of visualization, clustering and classification.

Acknowledgment

This research is wholly supported by the Hong Kong Government Competitive Earmarked Research Grant of project No. 9040803-570.

References

[1] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan, New York, 1994.

[2] G.E. Hinton, Special issue on connectionist symbol processing, Artif. Intell. 46 (1990) 1–2.

[3] J. L Elman, Distributed representations, simple recurrent networks, and grammatical structure, Mach. Learn. 7 (1991) 195–225.

[4] A. Sperduti, A. Starita, Dynamical neural networks construction for processing of labeled structures, Technical Report TR-95-1, University of Pisa, Dipartimento di Informatica, 1995.

[5] T. Plate, Holographic reduced representations, IEEE Trans. Neural Networks 6 (3) (1995) 623–641.

[6] T. Gärtner, J.W. Lloyd, P.A. Flach, Kernels for structured data, Mach. Learn. 57 (2004) 205–232.

[7] A. Küchler, C. Goller, Inductive learning in symbolic domains using structure driven recurrent neural networks, in: G. Gorz, S. Holldobler (Eds.), KI-96: Advances in Artificial Intelligence, Springer, Berlin, 1996, pp. 183–197.

[8] C. Goller, A. Küchler, Learning task-dependent distributed structure-representations by backpropagation through structure, in: IEEE International Conference on Neural Networks, 1996, pp. 347–352.

[9] A. Sperduti, A. Starita, Supervised neural networks for the classification of structures, IEEE Trans. Neural Networks 8 (3) (1997) 714–735.

[10] B. Hammer, Recurrent networks for structured data—a unifying approach and its properties, Cognitive Syst. Res. 3 (2) (2002) 145–165.

[11] J. Pollack, Recursive distributed representation, Artif. Intell. 46 (1–2) (1990) 77–106.

[12] A. Sperduti, Labeling RAAM, Connection Sci. 6 (4) (1994) 429–459.

[13] A. Sperduti, Encoding of labeled graphs by labeling RAAM, in: J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann, San Mateo, CA, 1994, pp. 1125–1132.

[14] A. Sperduti, A. Starita, C. Goller, Learning distributed representations for the classification of terms, Proceedings of the International Joint Conference on Artificial Intelligence, 1995, pp. 509–515.

[15] P. Frasconi, M. Gori, A. Sperduti, A general framework for adaptive processing of data sequences, IEEE Trans. Neural Networks 9 (5) (1997) 768–786.

[16] B. Hammer, Approximation capabilities of folding networks, in: M. Verleysen (Ed.), European Symposium on Artificial Neural Networks '99, D-facto Publications, 1999, pp. 33–38.

[17] B. Hammer, Generalization ability of folding networks, IEEE Trans. Knowl. Data Eng. 13 (2) (2001) 196–206.

[18] S.Y. Cho, Z. Chi, W.C. Siu, A.C. Tsoi, An improved algorithm for learning long-term dependency problem in adaptive processing of data structures, IEEE Trans. Neural Networks 14(4) (2003) 781–793.

[19] M. Hagenbuchner, Extension and evaluation of adaptive processing of structured information using artificial neural networks, Ph.D. Dissertation, Faculty of Informatics, University of Wollongong, 2002.

[20] S. Wu, M.K.M. Rahman, T.W.S. Chow, Content-based image retrieval using growing hierarchical self-organizing quadtree map, Pattern Recognition 38 (5) (2005) 707–722.

[21] T.W.S. Chow, M.K.M. Rahman, S. Wu, Content based image retrieval by using tree-structured features and multi-layer SOM, Pattern Anal. Appl. 9 (1) (2006) 1–20.

[22] S. Wu, T.W.S. Chow, Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density, Pattern Recognition 37 (2) (2004) 175–188.

[23] Z. Wang, M. Hagenbuchner, A.C. Tsoi, S.Y. Cho, Z. Chi,. Image classification with structured self-organization map, Proceedings of the 2002 International Joint Conference on Neural Networks, vol. 2, 2002, pp. 1918–1923.

[24] M. Hagenbuchner, A. Sperduti, A.C. Tsoi, A self-oganizing map for adaptive processing of structured data, IEEE Trans. Neural Networks 14 (3) (2003) 491–505.

[25] A.P. Azcarraga, Assessing self-organization using order metrics, Proceedings of the International Joint Conference on Neural Networks, vol. 6, National University of Singapore, IEEE, Piscataway, NJ, 2000, pp. 159–164.

[26] M. Halkidi, M. Vazirgiannis, Clustering validity assessment using multi representatives, Proceedings of SETN Conference, Thessaloniki, Greece, 2002.

[27] M.F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.

[28] G. Salton, C. Buckley, Term weighting approaches in automatic text retrieval, Technical Report TR87-881, Department of Computer Science, Cornell University, 1987, Inf. Process. Manage. 32(4) (1996) 431–443.

[29] P. Salembier, L. Garrido, Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval, IEEE Trans. Image Processing 9(4) (2000) 561–576.

[30] A. Sanfeliu, R. Alquézar, J. Andrade, J. Climent, F. Serratosa, J. Vergés, Graph-based representations and techniques for image processing and image analysis, Pattern Recognition 35 (2002) 639–650.

[31] J.Z. Wang, J. Li, G. Wiederhold, SIMPLIcity: semantics sensitive integrated matching for picture libraries, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 947–963.

[32] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, G. Taubin, The QBIC project: querying images by content using color, texture, and shape, Proceedings of SPIE—International Society of Optical Engineering, in: Storage and retrieval for image and video database 1908, 1993, pp. 173–187.

About the Author—M.K.M. RAHMAN received his B.Eng. degree in the Department of Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology in 2001. He is currently working towards the Ph.D. degree at City University of Hong Kong, Hong Kong. His research interests are structural data processing, neural network, pattern recognition and their applications.

About the Author—PIYANG WANG is now pursuing Ph.D. degree in the Department of Electronic Engineering of City University of Hong Kong. He was an exchange student from Shanghai Jiaotong University and obtained B.E. degree in City University of Hong Kong with first honor. His research interest areas are neural networks, pattern recognition, and their applications.

About the Author—TOMMY W.S. CHOW (IEEE M'93-SM'03) received the B.Sc. (First Hons.) and Ph.D. degrees from the University of Sunderland, Sunderland, U.K. He joined the City University of Hong Kong, Hong Kong, as a Lecturer in 1988. He is currently a Professor in the Electronic Engineering Department. His research interests include machine fault diagnosis, HOS analysis, system identification, and neural network learning algorithms and applications.

About the Author—SITAO WU received the B.E. and M.E. degrees in the department of electrical engineering of Southwest Jiaotong University, Chengdu, P.R. China in 1996 and 1999 respectively. He received the Ph.D. degree in the Department of Electronic Engineering of City University of Hong Kong, Hong Kong, P.R. China in 2004. His research interest areas are neural networks, pattern recognition and their applications.