

# Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery

Damminda Alahakoon, Saman K. Halgamuge, *Member, IEEE*, and Bala Srinivasan

**Abstract**—The growing self-organizing map (GSOM) has been presented as an extended version of the self-organizing map (SOM), which has significant advantages for knowledge discovery applications. In this paper, the GSOM algorithm is presented in detail and the effect of a spread factor, which can be used to measure and control the spread of the GSOM, is investigated. The spread factor is independent of the dimensionality of the data and as such can be used as a controlling measure for generating maps with different dimensionality, which can then be compared and analyzed with better accuracy. The spread factor is also presented as a method of achieving hierarchical clustering of a data set with the GSOM. Such hierarchical clustering allows the data analyst to identify significant and interesting clusters at a higher level of the hierarchy, and as such continue with finer clustering of only the interesting clusters. Therefore, only a small map is created in the beginning with a low spread factor, which can be generated for even a very large data set. Further analysis is conducted on selected sections of the data and as such of smaller volume. Therefore, this method facilitates the analysis of even very large data sets.

**Index Terms**—Clustering methods, hierarchical systems, knowledge discovery, neural networks, self-organizing feature maps, unsupervised learning.

## I. INTRODUCTION

WHEN humans try to make sense of complex problems, their natural tendency is to break the problem into smaller pieces, which, taken separately, would be easier to understand and solve [1]. A large data set may be of very high dimensionality and consist of such complex structure that even the most well planned data mining or analysis techniques might have difficulty extracting meaningful patterns from it. In many cases, the problem is not that of finding some patterns but of identifying the useful patterns from the large number that exist. An unsupervised technique such as cluster detection becomes useful in such situations.

Although data mining can be categorized into directed and undirected, there is a section of the data mining community that identifies data mining solely with undirected data mining [2]. The argument is that data mining is generally applied for unknown, unforeseen patterns, and therefore, if a hypothesis can be found using the labeled nature of directed data, it would not be data *mining*.

Automatic cluster detection when used as a data mining tool can properly be described as undirected knowledge discovery or unsupervised learning. Unsupervised learning methods play an important role in pattern recognition and knowledge acquisition problems. The function of unsupervised learning (or self-organizing) can be thought of as the development of a suitable clustering for a given data set. Therefore, the function of clustering algorithms in unsupervised learning problems is to identify an optimal partition in the data set. This can be further described as: find groupings in an unlabeled set of data vectors that share some well-defined mathematical or semantic similarities without any supervised learning procedure. Usually unsupervised clustering methods are applied when the classification of a given set of sample patterns is unknown. Therefore, this method is useful in data mining when it is attempted to find unforeseen or novel patterns in the data. But applying unsupervised algorithms to data that are already classified can also reveal interesting groupings, which were not identified earlier. Clustering is a useful tool when it is to deal with a large complex data set with many variables and unknown internal structure. In such situations, clustering would be the best tool to obtain an initial understanding of the structure inherent in the data. Thereafter (since a general idea of the data is now available), other data mining tools can be used to discover inter- and intracluster rules and patterns. The strengths of automatic cluster detection for data mining are:

- undirected technique;
- robust performance with diverse data types;
- easy to apply since there is no need to specify particular fields and inputs.

Therefore, clustering is a very useful tool when the analyst is faced with a large, complex data set containing many variables and a complicated unknown structure. At the beginning of a data mining operation, clustering would often be the best technique to use. Once automatic cluster detection has discovered regions of the data space that contain similar records, other data mining tools and techniques could be used to discover rules and patterns within the clusters.

The self-organizing map (SOM) has been used as a tool for mapping high-dimensional data into a two- (or three-) dimensional *feature map* [3]. It is then possible to visually identify the clusters from the map. The main advantage of such a mapping is that it would be possible to gain some idea of the structure of the data by observing the map, due to the *topology preserving* nature of the SOM. It has been theoretically proved that the SOM in its original form does not provide complete topology preservation, and several researchers in the past have attempted to overcome this limitation [4], [5].

Manuscript received July 15, 1999; revised January 11, 2000.

D. Alahakoon and B. Srinivasan are with the School of Computer Science and Software Engineering, Monash University, Caulfield East, Vic. 3145, Australia.

S. K. Halgamuge is with the Mechatronics Research Group, Department of Mechanical and Manufacturing Engineering, University of Melbourne, Parkville, Vic. 3052, Australia.

Publisher Item Identifier S 1045-9227(00)03710-3.

In most applications, the SOM has been used to map from high-dimensional input space to two dimensions. The usefulness of such a mapping for a given application will depend on how accurately it represents the input space. The SOM is normally represented as a two-dimensional (2-D) grid of nodes. When using the SOM, the size of the grid and the number of nodes have to be predetermined. The need for predetermining the structure of the network results in a significant limitation on the final mapping. It is often known only at the completion of the simulation that a different sized network would have been more appropriate for the application. Therefore, simulations have to be run several times on different sized networks to pick the optimum network [6]. A further limitation when using SOM for knowledge discovery occurs due to the user's not being aware of the structure present in the data. Therefore, it not only becomes difficult to predetermine the size of the network but it is also not possible to say when the map has organized into a proper cluster structure, as the user is not aware of the proper structure itself (in fact, finding the proper structure is one of the goals in data mining). The solution to this problem would be to determine the shape as well as the size of the network during the training of the network.

A dynamic feature map model called the growing self-organizing map (GSOM) is proposed in this paper. A detailed description of the GSOM algorithm is presented and its characteristics are justified. The other focus of this paper is the usefulness of the GSOM as a tool for the initial phase of data mining by identifying clusters in the data. The need for a measure for controlling the growth of the GSOM (or any such algorithm) is highlighted, and an indicator called the spread factor is presented as a method of achieving such a control. Finally, the usefulness of the spread factor is demonstrated with some experimental results on real data sets.

## II. SELF-GENERATING FEATURE MAPS FOR DATA MINING

The advantages of self-generating neural networks have caused some interest in the recent past. Several researchers have described such new neural architectures, both in the supervised and unsupervised paradigms. Although these new models obviously become more complex compared with their fixed structure counterparts, significant advantages could be gained from such dynamic models, the main advantage being their ability to grow (or change) structure (number of nodes and connections) to represent the application better. This becomes a very useful aspect in applications such as data mining, where it is not possible for the neural network designer to be aware of the inherent structure in the data.

There have been several extensive reviews [7], [8] on the supervised self-generating neural architectures. Our work so far has been discussed in several articles [9], [10], where a self-generating unsupervised feature map called the GSOM is presented. The potential of the GSOM for data mining (and analysis) was described in [11].

The rest of this paper is organized as follows. Section II-A highlights some previous attempts at developing a dynamic feature mapping model and also looks into the need of controlling the spread of such a model. Section III presents the GSOM al-

gorithm in detail with justifications. Section IV discusses the advantages of the GSOM compared with the traditional SOM, and Section V presents the concept of the spread factor and its usage to achieve useful hierarchical clustering for knowledge discovery. Section VI describes some experimental results to highlight the claims of the GSOM and the spread factor, and Section VII provides a summary discussion of this paper.

### A. Related Work in the Past

Several dynamic neural network models have been developed in the past, which attempt to overcome the limitations of the fixed structure networks. Recent work on such supervised models have been reported in [12] and [13]. Since the focus of this paper is on unsupervised self-generating neural networks, some of these models are considered in this section.

1) *Growing Cell Structures (GCS's)*: The GCS algorithm [14], [15] is based on the SOM, but the basic two-dimensional grid of the SOM has been replaced by a network of nodes whose connectivity defines a system of triangles. The GCS starts with a triangle of cells at random positions in  $R^n$ . This triangle is distributed as well as possible over the area of nonzero probability in the space. Heuristics are used to both add and remove network nodes and connections. The algorithm results in a network graph structure  $G = (V, E)$ , where  $V$  is the set of nodes and  $E$  is the set of connections between them. The GCS uses a drawing method, which works well with relatively low-dimensional data, but the mapping cannot be guaranteed to be planar for high-dimensional data. This causes problems in visualizing high-dimensional data.

2) *Neural Gas Algorithm*: Martinetz and Shulten developed the Neural Gas algorithm [16], which can also be categorized as an unsupervised self-generating neural network. The network starts with no connections and a fixed number of units floating in the input vector space. When the inputs are presented to the network, units are adapted and connections are created between the winning units and the closest competitor. There is also a mechanism for aging and removal of units. Neural Gas uses a fixed number of units, which have to be decided prior to training. This again results in the same limitations as the SOM in data mining applications. The dimensionality of the Neural Gas depends on the respective locality of the input data. Therefore, the network can develop different dimensionality for different sections, which can result in visualization difficulties.

3) *Incremental Grid Growing (IGG)*: IGG [17] builds the network incrementally by dynamically changing its structure and connectivity according to the input data. IGG network starts with a small number of initial nodes and generates nodes from the boundary of the network using a growth heuristic. Connections are added when an internode weight difference drops below a threshold value and connections are removed when weight differences increases. Adding nodes only at the boundary allows the IGG network to always maintain a two-dimensional structure, which results in easy visualization. Therefore, the structure of the data is apparent in the structure of the network without having to plot the weight values.

The GSOM algorithm presented in the next section has some characteristics similar to the IGG but uses a weight initialization method, which is simpler and reduces the possibility of

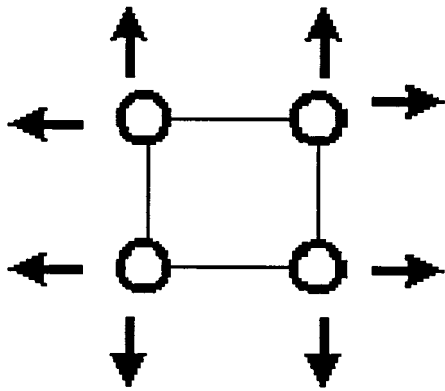


Fig. 1. Initial GSOM.

twisted maps. The initial GSOM is shown in Fig. 1. The GSOM also carries out node generation interleaved with the self-organization, thus letting the new nodes smoothly join the existing network. The Space Partition Network (SPAN) algorithm [18] and the Morphogenetic algorithm [19] are other attempts at dynamically creating SOM's to reduce the limitations of the fixed structure networks. The SPAN algorithm generates new neurons as an *adaptive vector quantizer* starting from an initial network of zero neurons. Heuristics are defined for neuron generation and annihilation. The Morphogenetic algorithm generates whole rows or columns from a grid where a new neuron is required or needs to be removed, thus always maintaining a rectangular grid structure.

### B. The Need for Controlling the Growth in Self-Generating Maps

When using feature maps to identify the clusters, it is advantageous if there is a mechanism for initially observing the most significant clusters and then, once the data analyst has some idea of the overall data set, to further spread out the mapping and obtain finer clusters. This will also facilitate the data analyst in making decisions on *regions* of the data that are not of interest and direct the finer clustering only to regions of interest. When a set of clusters has been identified, a data analyst sometimes needs to have a look at the map of only selected attributes (dimensions) of the data to gain further insight into the structure of the data. The same spread of the map (or a controlled spread) has to be achieved in this further analysis for the analyst to compare the results and arrive at a fair opinion. It is also necessary to have a measure for controlling the spread of the map such that finer clustering of the data set can be achieved hierarchically as required by the data analyst and on the regions of interest.

To achieve this control on the spread of the map, a method is developed for the analyst to specify the amount of spread required by specifying a *spread factor*. It is anticipated that a low spread factor will be given at the beginning of analysis and then gradually increasing the spread factor value for further observations of selected regions in the data. The spread factor takes values between zero and one and is independent of the number of dimensions in the data. It is possible to compare the results of different data sets with a different number of attributes by mapping them with the same spread factor.

### III. GROWING SELF-ORGANIZING MAP ALGORITHM

The GSOM is an unsupervised neural network, which is initialized with four nodes and *grows* nodes to represent the input data [10], [11]. During the node growth, the weight values of the nodes are *self-organized* according to a similar method as the SOM.

The GSOM process is as follows.

- 1) Initialization phase.
  - a) Initialize the weight vectors of the starting nodes (usually four) with random numbers.
  - b) Calculate the growth threshold (GT) for the given data set according to the user requirements.
- 2) Growing phase.
  - a) Present input to the network.
  - b) Determine the weight vector that is closest to the input vector mapped to the current feature map (winner), using Euclidean distance (similar to the SOM). This step can be summarized as: find  $q'$  such that  $|v - w_{q'}| \leq |v - w_q| \forall q \in \mathcal{N}$ , where  $v, w$  are the input and weight vectors, respectively,  $q$  is the position vector for nodes, and  $\mathcal{N}$  is the set of natural numbers.
  - c) The weight vector adaptation is applied only to the neighborhood of the winner and the winner itself. The neighborhood is a set of neurons around the winner, but in the GSOM the starting neighborhood selected for weight adaptation is smaller compared to the SOM (localized weight adaptation). The amount of adaptation (learning rate) is also reduced exponentially over the iterations. Even within the neighborhood, weights that are closer to the winner are adapted more than those further away. The weight adaptation can be described by
$$w_j(k+1) = \begin{cases} w_j(k), & j \notin N_{k+1} \\ w_j(k) + LR(k) \times (x_k - w_j(k)), & j \in N_{k+1} \end{cases}$$
where the learning rate  $LR(k)$ ,  $k \in \mathcal{N}$  is a sequence of positive parameters converging to zero as  $k \rightarrow \infty$ .  $w_j(k)$ ,  $w_j(k+1)$  are the weight vectors of the node  $j$  before and after the adaptation, and  $N_{k+1}$  is the neighborhood of the winning neuron at  $(k+1)^{th}$  iteration. The decreasing value of  $LR(k)$  in the GSOM depends on the number of nodes existing in the network at time  $k$ .
  - d) Increase the error value of the winner (error value is the difference between the input vector and the weight vectors).
  - e) When  $TE_i \leq GT$  (where  $TE$  is the total error of node  $i$  and  $GT$  is the growth threshold). Grow nodes if  $i$  is a boundary node. Distribute weights to neighbors if  $i$  is a nonboundary node.
  - f) Initialize the new node weight vectors to match the neighboring node weights.

- g) Initialize the learning rate ( $LR$ ) to its starting value.
  - h) Repeat steps b)–g) until all inputs have been presented and node growth is reduced to a minimum level.
- 3) Smoothing phase.
- a) Reduce learning rate and fix a small starting neighborhood.
  - b) Find winner and adapt the weights of winner and neighbors in the same way as in growing phase.

The GSOM adapts its weights and architecture to represent the input data. Therefore, in the GSOM, a node has a weight vector and two-dimensional coordinates that identify its position in the net, while in the SOM the weight vector is also the position vector.

The GSOM has two modes of activation:

- 1) training mode or generating mode;
- 2) testing mode or querying mode.

The training mode consists of the three phases described above, and the testing mode is run to identify the positions of a set of inputs within an existing (trained) network. This can be regarded as a calibration phase if known data are used. For unclassified data the *closeness* of new inputs to the existing clusters in the network can be measured. In the following section, these phases are described in detail.

#### A. Initialization Phase

The network is initialized with four nodes because:

- 1) it is a good starting position to implement a two-dimensional lattice structure;
- 2) all starting nodes are boundary nodes (definition of boundary node given in Section III-B); thus each node has the freedom to grow in its own direction at the beginning.

The starting four nodes are initialized with random values from the input vector space, or the input vector value range. Since the input vector attributes are normalized to the range 0–1, the initial weight vector attributes can take random values in this range. Therefore, no restrictions are enforced on the directions of the lattice growth. Thus the initial square shape lets the map grow in any direction solely depending on the input values.

A numeric variable  $H_{\text{Err}}$  is initialized to “0” at initialization. This variable will keep track of the highest accumulated *error* value in the network. A value called the spread factor (SF) has to be specified. The SF allows the user (data analyst) to control the growth of the GSOM and is independent of the dimensionality of the data set used. SF is used by the system to calculate the growth threshold (GT). This will act as a threshold value for initiating node generation. A high GT value will result in less spread out map, and a *low* GT will produce a well-spread map.

#### B. Growing Phase

The set of neuron weight vectors  $W_i$  in the GSOM can be considered as a vector quantization of the input space so that each neuron  $i$  is used to represent a region  $V_i$  in which all points are closer to  $W_i$  than the weight vector of any other neuron. Therefore,  $V_i$  can be considered as a Voronoi region [20]. In

other words, the weight vectors partition the input space into Voronoi regions, with each region represented by one neuron. In the growing phase, when input data are presented to the network, a *winner* is found as per the algorithm above. It was described that the difference between the input vector and the corresponding weight vector of the winner is accumulated as error value for that neuron. If neuron  $i$  contributes significantly to the total error (distortion) of the network, then its Voronoi region is said to be under represented by the neuron  $i$ . Therefore, a new neuron is generated as a neighbor of neuron  $i$  to achieve a better representation of the region to determine the criteria for new node generation. The following behavior is considered during training of a feature map.

- 1) If the neural network has enough neurons to process the input data, then during training, the weight vectors of the neurons are adapted such that the distribution of the weight vectors will represent the input vector distribution.
- 2) If the network has insufficient neurons, a number of input vectors, which otherwise would have been spread out to neighboring neurons, will be accumulated on a single neuron.

Therefore, a measure called the error distance ( $E$ ) is defined

$$E_i(t+1) = E_i(t) + \sqrt{\sum_{k=1}^{\text{Dim}} \text{Met}(v_k, w_{i,k})^2} \quad (1)$$

for neuron  $i$  at time (iteration)  $t$  and where  $Dim$  is the dimensions (attributes) in the input data, and  $v, w$  are input and weight vectors, respectively.  $\text{Met}$  is a metric that measures the distance between the vectors  $v$  and  $w$ . Thus for each winner node, the difference between the weight vector and the input vector is calculated as an *error* value. This value is accumulated over the iterations if the same node wins on several occasions. Using square of the Euclidean distance as the metric

$$E_i(t+1) = E_i(t) + \sqrt{\sum_{k=1}^{\text{Dim}} (v_k - w_{i,k})^2}. \quad (2)$$

At each weight updating

- if  $E_i^{\text{new}} > H_{\text{Err}}$  then  $H_{\text{Err}} = E_i^{\text{new}}$ ;
- else  $H_{\text{Err}}$  remains unchanged.

Thus  $H_{\text{Err}}$  will always maintain the largest error value for a neuron in the network. The error value calculated for each node can be considered as a quantization error and the total quantization error would be

$$QE = \sum_{i=1}^N E_i \quad (3)$$

where  $N$  is the number of neurons in the network and  $E_i$  is the error value for neuron  $i$ . The total quantization error  $QE$  is used as a measure of determining when to generate a new neuron. If a neuron  $i$  contributes substantially toward the total quantization error, then its Voronoi region  $V_i$  in the input space is said to be underrepresented by neuron  $i$ . Therefore, a new neuron is created to share the load of neuron  $i$ .

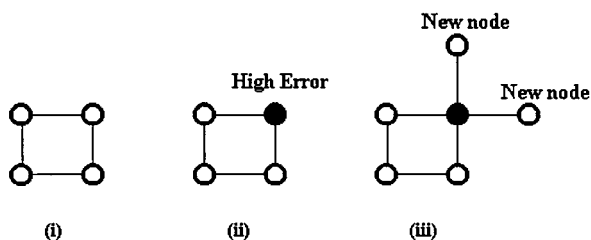


Fig. 2. New node generation from the boundary of the network.

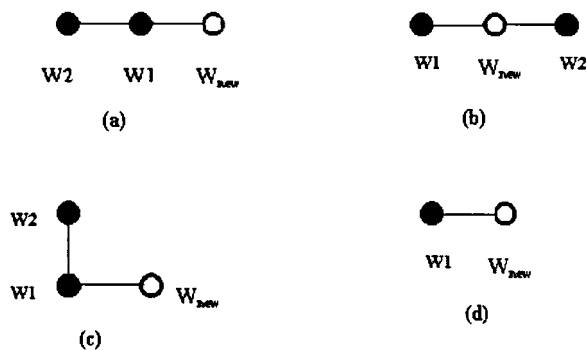


Fig. 3. Weight initialization of new nodes.

A neuron  $i$  should generate another neuron if

$$\frac{\partial QE}{\partial E_i} E_i > GT. \tag{4}$$

Since in the implementation  $H_{Err}$  contains the largest error distance for a neuron

- if  $H_{Err} > GT$ , then grow new nodes.

1) *New Node Generation*: New nodes will always be grown from a boundary node (Fig. 2). A boundary node is one that has at least one of its immediate neighboring positions free of a node. In our model, each node will have four immediate neighbors. Thus a boundary node can have from one to three neighboring positions free. If a node is selected for growth, all its free neighboring positions will be grown new nodes. New nodes are generated on all free neighboring positions, as this is computationally easier to implement than calculating the exact position of the new node. This will create some redundant (dummy) nodes, but dummy nodes can be easily identified and removed after a few iterations, as they will accumulate “0” hits.

2) *Weight Initialization of New Nodes*: The newly grown nodes will be assigned initial weight values. Since the older nodes would be at least partly organized at this stage (organization of the existing nodes happens from the start), random initialization of the new nodes will introduce weight vectors that do not match their neighborhoods. Therefore, the following method is used considering the smoothness properties of the existing map and thus initializing the new weights to match their neighborhoods. For the weight initialization of new nodes, there are four situations to consider. Fig. 3 shows the different cases of a node initiating the growth of a new node.  $w_1$  and  $w_2$  are weights of the growth initiating node and one of its neighbors, respectively.  $w_{new}$  is the weight of the newly generated node. Since the GSOM begins with four initial nodes, and a new node is always generated as a neighbor of an existing node, cases shown

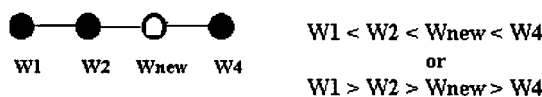


Fig. 4. Flow of the weight vectors.

in Fig. 3(a) and (b) are the most frequently occurring situations. Case (b) can occur in a very rare situation where the GSOM will *branch back* after spreading outwards. In such a situation, case (b) will be given priority over case (a) and (c) [since when applicable, case (b) provides a better estimate for the weight with an *in between* value]. Case (d) is used, when node elimination removes all the neighbors of a node, and such a node initiates growth (node elimination is on going work).

- The new node has two consecutive old nodes on one of its sides [Fig. 3(a)].  
 if  $w_2 > w_1$   
 then  $w_{new} = w_1 - (w_2 - w_1)$   
 if  $w_1 > w_2$   
 then  $w_{new} = w_1 + (w_1 - w_2)$ .
- The new node is in between two older nodes [Fig. 3(b)]

$$w_{new} = \frac{w_1 + w_2}{2}.$$

- The new node has only one direct neighbor (the parent) older node. But the older node has a neighbor on one side which is not the side directly opposite the new node [Fig. 3(c)].  
 if  $w_2 > w_1$   
 then  $w_{new} = w_1 - (w_2 - w_1)$   
 if  $w_1 > w_2$   
 then  $w_{new} = w_1 + (w_1 - w_2)$ .
- The new node has only one neighboring older node. This can occur at the beginning of growth or when a node that has become isolated due to node removing (on going work) initiates growth again [Fig. 3(d)].  $w_{new} = m$ , where  $m = (r_1 + r_2)/2$  and  $r_1, r_2$  being the lower and upper values of the range of the weight vector distribution.

In cases (a)–(c), if  $r_1 \leq w_{new} \leq r_2$ , the new node will be assigned a weight value, which will merge with the flow of the neighboring weights, i.e., in Fig. 4,  $w_1 < w_2 < w_{new} < w_4$ .

In the case of (d) or if  $w_{new}$  is out of the range  $[r_1, r_2]$ , then  $w_{new}$  is assigned the value  $(r_1 + r_2)/2$ . The justification for this is: if a satisfactory value could not be found with (a)–(c), the node will be assigned a mid-range value, e.g., in case of binary data this value would be 0.5. The self-organization would then increase or decrease this value as necessary to fit the neighborhood. This method is justified as: if the cases (a)–(c) can be used, then they will produce a weight value closer to the flow of the neighborhood weights. If it is not the case, then the mid-range value should be used, letting the self-organization make the adaptation.

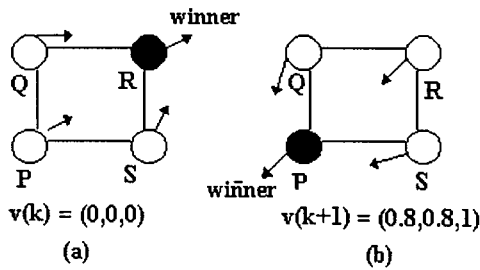


Fig. 5. Weight fluctuation at the beginning due to high learning rates.

### C. Smoothing Phase

The smoothing phase occurs after the new node growing phase in the training mode. The growing phase stops when new node growth saturates, which can be identified by the low frequency of new node growth. Once the node growing phase is complete, the weight adaptation is continued with a lower rate of adaptation. No new nodes are added during this phase. The purpose is to smooth out any existing quantization error, especially in the nodes grown at the latter stages of the growing phase.

During the smoothing phase, inputs to the network are similar to those of the growing phase. The starting learning rate ( $LR$ ) in this phase is less than in the growing phase, since the weight values should not fluctuate too much without converging. The weight fluctuation at the beginning is shown in Fig. 5. The input data are repeatedly entered to the network until convergence is achieved. The smoothing phase is stopped when the error values of the nodes in the map become very small.

Therefore, the smoothing phase has the following differences from the growing phase.

- 1) The learning rate ( $LR$ ) is initialized at a lesser value.
- 2) The neighborhood for weight adaptation is constrained only to the immediate neighborhood (even smaller than in the growing phase).
- 3) The rate of learning rate depreciation is smaller.
- 4) No node growth.

## IV. ADVANTAGES OF GSOM OVER OTHER SIMILAR WORK

### A. Learning Rate Adaptation

The learning rate adaptation of the GSOM was described in Section III. The weight adaptation rule used can be written as

$$LR(t+1) = LR(t) \times \alpha$$

where  $\alpha$  is the learning rate reduction and is implemented as a constant value  $0 < \alpha < 1$ , where  $LR$  is the learning rate at the  $t$ th iteration. The use of  $\alpha$  in the equation makes  $LR(t)$  converge to zero as  $t \rightarrow \infty$ .  $LR$  is first initialized with a *high* value, similar to the SOM. But this causes a problem in the GSOM due to the small number of nodes at the beginning.

Due to the small number of nodes at the beginning, the same nodes would be selected as the winners neighborhood for very *different* input vectors. This will cause the weights of the same set of nodes to fluctuate in completely different directions. This situation can be improved when the input data are ordered according to attribute values, since the ordered data vectors will

make the map grow in a certain direction. Therefore, by the time a different type of input is presented, the map would be sufficiently grown, such that all nodes will not fluctuate toward the new input. Although this method is practical with a *known*, small data set, it would be impossible to order an unknown set of data since the data analyst does not know the relationships between the attributes.

As a solution, a new learning rate reduction rule, which takes the number of current nodes in the neural network into consideration, is introduced. The learning rate reduction rule can now be stated as

$$LR(t+1) = \alpha \times \psi(n) \times LR(t) \quad (5)$$

where  $\psi(n)$  is a function of  $n$ , the number of current nodes in the map, and is used to manipulate the  $LR$  value such that  $LR(t)$ , which is highest at the initial  $t$  values, is then reduced according to the number of current nodes; i.e.,  $\psi(n)$  is a function that gradually takes higher values as the map grows and the number of nodes becomes larger. One simple formula that can be used is  $(1 - R/n(t))$ . In the experiments described in this paper,  $R = 3.8$  is used arbitrarily since the starting number of nodes is four. The new formula will force the following functionality on the weight adjustment algorithm.

- 1) At the initial stage of growth, when the number of nodes are few, the high  $LR$  values will be reduced with the  $\psi(n)$ . This will result in reducing the fluctuations of the weights of a small map.
- 2) As the network grows, the  $\psi(n)$  will take gradually higher values, which will result in the  $LR$ 's not being reduced as much as in the situation described in (1). Since the map is now larger, we require the high  $LR$  to *self-organize* the weights within the regions of the map (the high  $LR$  will not affect the other regions at this stage due to the map's being larger, and the other regions will not belong to the same neighborhood).

Therefore, the modified formula provides the GSOM with the required weight adjustment, which results in providing a better organized set of weight values, which will then be further smoothed out during the smoothing phase.

### B. Localized Neighborhood Weight Adaptation

During an SOM training, the neighborhood (for weight adaptation) is large at the beginning and can shrink linearly to one node during the ordering stage. The *ordering* of the weight vectors  $w_i$  occurs during this initial period, while the remaining steps are only needed for the fine adjustment (convergence) of the map.

The GSOM does not require an ordering phase, since new weight nodes are initialized to fit in with the existing neighborhood weights but requires repeated passes over a small neighborhood where the neighborhood size reduces to unity. This starting fixed neighborhood can be considered very small compared to SOM. Therefore, during the growing phase, the GSOM initializes the learning rate and the neighborhood size  $N_k$  to a starting value at each new input. Weight adaptation is then carried out with reducing neighborhood and learning rate until neighborhood is unity, and initializes again for the next new

input. Weight updating in overlapping neighborhoods (overlapping for different inputs) will not create disorder in the map since, the weight adaptation in the GSOM can be considered as

$$\begin{aligned} dw_i/dt &= LR \times (v - w_i) & \text{for } i \in N_k \\ &= 0 & \text{otherwise.} \end{aligned}$$

An ordered set of weights  $w_1, w_2, \dots, w_n$  cannot be disordered by the adaptation caused by the above equation, since if all partial sequences are ordered, then it cannot change the relative order of any pair  $(w_i, w_j), i \neq j$ . Therefore, once the weights are ordered (by initialization and self-organization), further weight adaptation will not create disorder.

### C. Error Distribution of Nonboundary Nodes

The GSOM generates nodes only from the boundary of the network. The advantage of this is that the resulting network is always a two-dimensional grid structure, which is easier to visualize. A limitation of this method arises when the *error* value of a nonboundary node exceeds the GT due to high-density areas in the data. This occurs due to the inability of the nonboundary node to generate new nodes. Therefore, the resulting map may not give a proportionate representation of the input data distribution. Such proportionate representation has been called the *magnification factor* [3]. In biological brain maps, the areas allocated to the representation of various sensory features are often believed to reflect the importance of the corresponding feature sets. Although no attempt is made to justify the GSOM with biological phenomena, it was found that such proportional representation of the frequency of occurrence by the area in a feature map would be useful for a data analyst to immediately identify regions of high frequency, thus giving some idea about the distribution of the data. It was also seen that other than for the very simple data sets, the GSOM will stop growing when a certain amount of spread has been achieved.

A benchmark data set of 99 animals (the zoo or animal data base) [21], is used to demonstrate the result of the error distribution function. The data are 16-dimensional (16 attributes), and further details of this data set are given in Section VI. It can be seen from Fig. 6 that the map gives a very congested view of the data. This is due to the fact that growth can only occur from the boundary of the map. Fig. 7 shows the GSOM for the same data after applying the weight distribution function. Due to this modification, Fig. 7 gives a better spread of the data set.

In the SOM, this situation is left to be handled by the self-organization process, and a sufficient number of iterations will result in the map's spreading out, if a large enough grid had been initially created. In the GSOM, since the requirement is to grow nodes as and when required, there has to be a mechanism to initiate growth from the boundary for the map to spread out. The following error distribution criteria have been implemented to handle this growth of the map.

It was described above that a node will be selected to initiate growth when its *error* exceeds the growth threshold GT. When such a node is not in the boundary of the network, it cannot grow

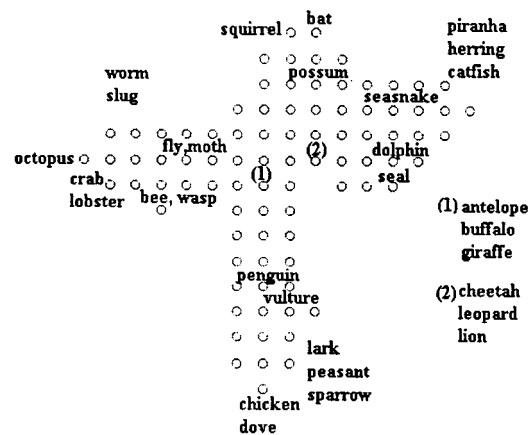


Fig. 6. The zoo data set mapped to the GSOM without the weight distribution.

new neighbors due to its position. The following formula is used to *distribute* the error to the neighboring nodes:

$$E_{t+1}^w = GT/2 \quad (6)$$

where  $E_t^w$  is the error value of the winner and GT is the growth threshold. The error value of the immediate neighbors of the winner are increased as

$$E_{t+1}^{n_i} = E_t^{n_i} + \gamma E_t^w \quad (7)$$

where  $E_{t+1}^{n_i}$  ( $i$  can be 1–4) is the error value of the  $i$ th neighbor of the winner  $E_t^w$  and  $\gamma$  is a constant value called the *factor of distribution (FD)*, which controls the increase in the error accumulation. The *FD* values used in the experiments are  $0 < \gamma < 1$ .

Therefore, by using (6), the error value of the high error node is reduced to half the growth threshold. With (7), the error value of the immediate neighboring nodes is increased; therefore, the two equations produce an effect of spreading the error outwards from the high error node. This type of spreading out will in time (iterations) ripple outwards and cause a boundary node to increase its error value. Therefore, the purpose of (6) and (7) is to give the nonboundary nodes some ability in initiating (although indirectly) node growth.

## V. KNOWLEDGE DISCOVERY BY HIERARCHICAL CLUSTERING OF THE GSOM

### A. The Spread-Out Factor (SF)

As described in Section III, the GSOM uses a threshold value called the GT to decide when to initiate new node growth. GT will decide the amount of spread of the feature map to be generated. Therefore, if only an abstract picture of the data is required, a large GT will result in a map with a fewer number of nodes. Similarly, a smaller GT will result in the map's spreading out more. When using the GSOM for data mining, it might be a good idea to first generate a *smaller* map, only showing the most significant clusters in the data, which will give the data analyst an overall picture of the inherent clustering in the total data set.

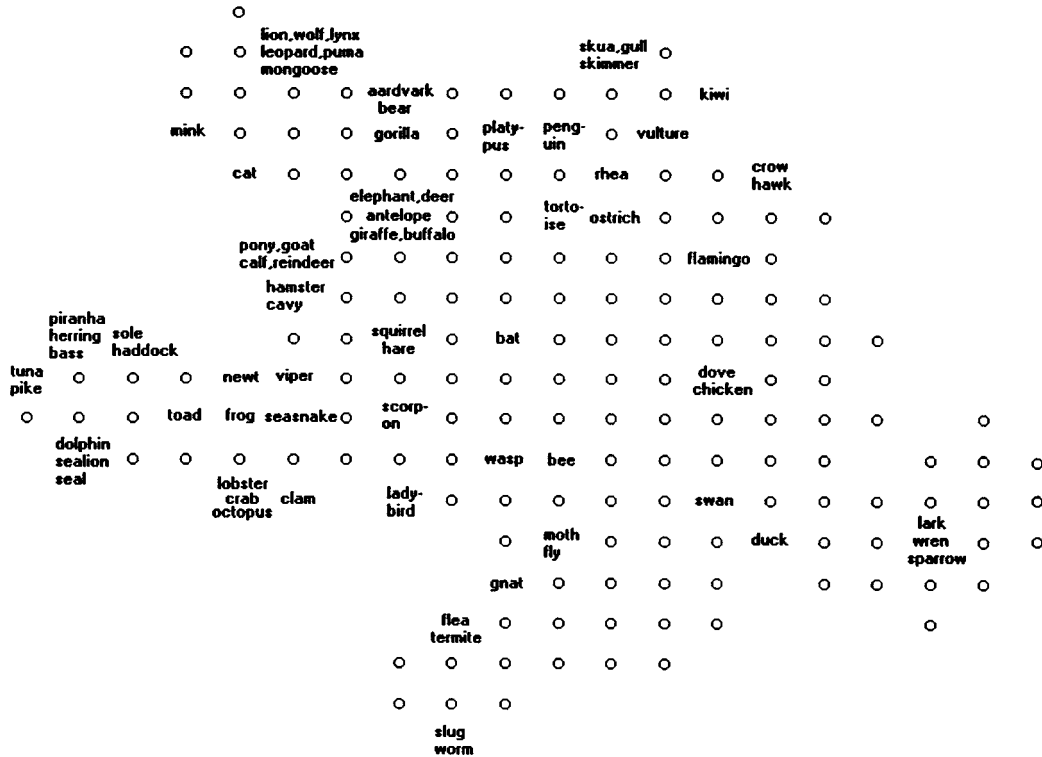


Fig. 7. The zoo data set mapped to the GSOM with the weight distribution.

The node growth in the GSOM is initiated when the error value of a node exceeds the GT. The total error value for node  $i$  is calculated as

$$TE_i = \sum_{H_i} \sum_{j=1}^{\mathcal{D}} (x_{i,j} - w_j)^2 \quad (8)$$

where  $H_i$  is the number of hits to the node  $i$  and  $\mathcal{D}$  is the dimension of the data.  $x_{i,j}$  and  $w_j$  are the input and weight vectors of the node  $i$ , respectively. For a boundary node to grow a new node, it is required that

$$TE_i \geq GT. \quad (9)$$

The GT value has to be experimentally decided depending on the requirement for the map growth. As can be seen from (8), the dimension of the data set will make a significant impact on the accumulated error ( $TE$ ) value, and as such will have to be considered when deciding the GT for a given application.

Since  $0 \leq x_{i,j}, w_j \leq 1$ , the maximum contribution to the error value by one attribute (dimension) of an input would be

$$\max |x_{i,j} - w_j| = 1.$$

Therefore, from (8)

$$TE_{\max} = D \times H_{\max} \quad (10)$$

where  $TE_{\max}$  is the maximum error value and  $H_{\max}$  is the maximum possible number of hits. If  $H(t)$  is considered to be the number of hits at time (iteration)  $t$ , the GT will have to be set such that

$$0 \leq GT < D \times H(t). \quad (11)$$

Therefore, GT has to be defined based on the requirement of the map spread. It can be seen from (11) that the GT value will depend on the dimensionality of the data set as well as the number of hits. Therefore, it becomes necessary to identify a different GT value for data sets with different dimensionality. This becomes a difficult task, especially in applications such as data mining, since it is necessary to analyze data with different dimensionality as well as the same data under different attribute sets. It also becomes difficult to compare maps of several data sets since the GT cannot be compared over different data sets. Therefore, the user definable parameter is introduced. The SF can be used to control and calculate the GT for GSOM's, without the data analyst's having to worry about the different dimensions.

The growth threshold can be defined as

$$GT = D \times f(SF)$$

where  $SF \in R$ ,  $0 \leq SF \leq 1$ , and  $f(SF)$  is a function of SF, which is identified as follows.

The total error  $TE_i$  of a node  $i$  will take the values

$$0 \leq TE_i \leq TE_{\max} \quad (12)$$

where  $TE_{\max}$  is the maximum error value that can be accumulated. This can be written as

$$0 \leq \sum_H \sum_{j=1}^{\mathcal{D}} (x_{i,j} - w_j)^2 \leq \sum_{H_{\max}} \sum_{j=1}^{\mathcal{D}} (x_{i,j} - w_j)^2. \quad (13)$$

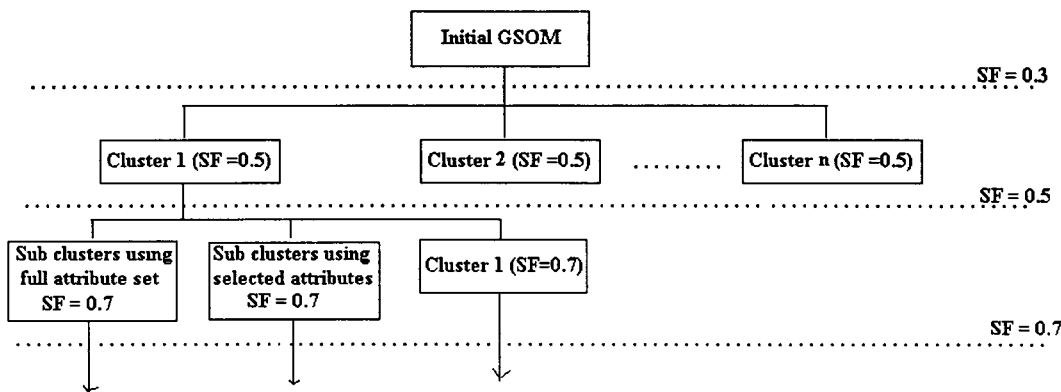


Fig. 8. The different options available to the data analyst using GSOM for hierarchical clustering of a data set.

Since the purpose of the GT is to let the map grow new nodes by providing a threshold for the error value, and the minimum error value is zero, it can be argued that for growth of new nodes

$$0 \leq GT \leq \sum_{H_{max}} \sum_{j=1}^D (x_{i,j} - w_j)^2. \quad (14)$$

Since the maximum number of hits ( $H_{max}$ ) can theoretically be infinite, (14) becomes  $0 \leq GT \leq \infty$ . According to the definition of spread factor, it is necessary to identify a function  $f(SF)$  such that

$$0 \leq SF \leq 1$$

and

$$0 \leq D \times f(SF) \leq \infty.$$

In other words, a function  $f(x)$  that takes the values zero to  $\infty$ , when  $x$  takes the values zero to one, is to be identified.

A Napier logarithmic function of the type  $y = -a \times \ln(1-x)$  is one such equation that satisfies these requirements. If  $\eta = 1 - SF$  and

$$GT = -D \times \ln(1 - \eta)$$

then

$$GT = -D \times \ln(SF).$$

Therefore, instead of having to provide a GT, which would take different values for different data sets, the data analyst can now provide a value SF, which will be used by the system to calculate the GT value depending on the dimensions of the data. This will allow the GSOM's to be identified with their spread factors and can form a basis for comparison of different maps.

The graph in Fig. 9 shows how the GT value of data sets with different dimensions change according to a given spread factor value.

So far, an equation is derived to calculate the growth threshold for a GSOM from a given spread factor. The work on dynamic feature maps in the past has been concentrated on obtaining an accurate topographical mapping of the data. But it is essential for knowledge discovery applications to have some control on the growth (or spread) of the map. This can be achieved with the spread factor.

- 1) Since the spread factor takes values from zero to one, where zero is the least spread and one is the maximum spread, the data analyst will be able to specify the amount of spread required. Generally, for knowledge discovery applications, if no previous knowledge of the data exists, it would be a good idea to initially use a low spread factor (between 0–0.3). This will produce a GSOM that highlights the most significant clusters. From these initial clusters, the analyst can decide whether it is necessary to study a further spread-out version of the data. Else, the analyst can select the regions of the map (or clusters) that are of interest and generate GSOM's on the selected regions using a larger SF value. This will allow the analyst to do a finer analysis of the areas of interest, which have now been separated from the total data set. Fig. 10 shows how this type of analysis can produce an incremental hierarchical clustering of the data.
- 2) During cluster analysis, it may be necessary (and useful) for the analyst to study the effect of removing some of the attributes (dimensions) on the existing cluster structure. This might be useful in confirming opinions on non-contributing attributes on the clusters. The spread factor facilitates such further analysis since it is independent of the dimensionality of the data. This is very important, as the growth threshold depends on the dimensionality.
- 3) When comparing several GSOM's of different data sets, it would be useful to have a measure for denoting the spread of the maps. Since the dimensions of the data sets could be different, the spread factor (independent of dimensionality) can be used to identify the maps by the amount of spread, across different data sets. This also opens up very useful opportunities for an organization that wants to automate this process of hierarchical clustering. The system can be configured to start clustering with  $SF = a$  and gradually continue until  $SF = x$ , where  $a < x$ .

Fig. 8 shows the different options available to the data analyst with the GSOM, due to the control achieved by the SF. The figure shows the initial GSOM generated with SF = 0.3. The clusters are then identified and expanded at a higher level of SF (0.5). At each level, the data analyst has the choice of expanding the whole map or expanding selected clusters either with their full set of attributes or using only a selected subset of

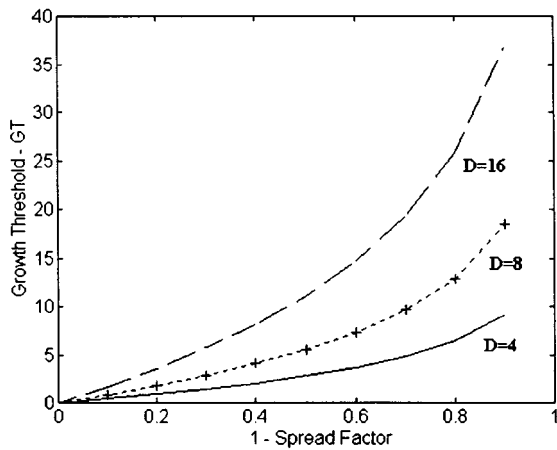


Fig. 9. Change of GT values for data with different dimensionality ( $D$ ) according to the spread factor.

the attributes. In most cases, the data analyst will use a combination of these methods to obtain an understanding of the data.

## VI. EXPERIMENTAL RESULTS

This section describes how the GSOM is applied to a number of data sets, and the results have been presented. The data set used for the first two experiments is the zoo data base. There are a total of 18 attributes available, which take values as shown for one animal (lion) in Table I.

The type of animal is specified by a numeric value from 1 to 7. The name and the type are removed from those attributes, and the GSOM is generated and trained with the remaining 16 attributes. Several experiments are described with the zoo data set to demonstrate the functionality of the GSOM.

### A. The Spread of the GSOM with Increasing SF Values

The results of the first experiment are shown in Figs. 11 and 12. Fig. 11 shows the GSOM of the zoo data with a low (0–1) spread factor, while in Fig. 12, an SF value of 0.85 is used. In Fig. 11, it can be seen that the different types of animals have been grouped together, and also similar subgroups have been mapped near each other. An SF around 0.3 would have given a better separated picture of the clusters. But it is still possible to see that there are three to four main groupings in the data set. One of the main advantages of the GSOM over the SOM is highlighted with this figure. The GSOM indicates the groupings in the data by its shape even when generated with a low SF. Fig. 11 has branched out in three directions, indicating the three main groups in the data (mammals, birds, and fish). The *insects* have been grouped together with some other animals but this group is not shown due to the low SF value.

Fig. 12 shows the same data set mapped with a higher (0.85) spread factor. It is possible to see the clusters clearly, as they are now spread out further. The clusters for *birds*, *mammals*, *insects*, and *fish* have been well separated. Since Fig. 12 is generated with a high SF value, even the subgroupings have appeared in this map. The *predatory birds* have been separated into a separate subcluster from other birds. The other subgroupings of birds can be identified as *airborne* and *nonairborne* (chicken, dove), and *aquatic*. The flamingo has been completely separated due to

its being the only *large* bird in the selected data. *The mammals* have been separated into *predators* and *nonpredators*, and the nonpredators have been separated into *wild* and *domestic* subgroups.

With this experiment it can be seen that the SF controls the spread of the GSOM. An interesting observation from this experiment is the way the GSOM can *branch out* to represent the data set. Due to this flexible shape of the network, the GSOM can represent a set of data with a fewer number of nodes (at the equal amount of spread) compared with the SOM. This has been proved experimentally by extensive testing on different data sets, the results of which were reported in previous work [9], [10], [22]. This becomes a significant advantage when training a network with a very large data set, since the reduction in the number of nodes will result in a reduction in processing time and also less computer resources.

### B. Hierarchical Clustering of Interesting Clusters

Although the complete map can be expanded with a higher SF value for further analysis, as shown in Fig. 12, it would sometimes be advantageous to select and expand only the areas of interest, as shown in Fig. 8. The data analyst can therefore continue hierarchically clustering the selected data until a satisfactory level of clarity is achieved. Fig. 13 shows the hierarchical clustering of the same data set used in the previous experiment. The upper right corner of Fig. 13 shows a section of the GSOM from Fig. 11. It is assumed that the analyst has selected two clusters (as shown with circles) and requires a more detailed view of the clusters. Two separated GSOM's have been generated for the selected clusters, using a higher (0.6) SF value. The subclustering inside the selected cluster is now clearly visible. The nondomestic/nonpredatory mammals have been mapped all together, which is due to all of them's having the same attribute values. The predators and the domestic mammals have been separated. The reasons for the other separations can also be investigated, e.g., it might be important to find out why *bear* has been separated from the other predatory mammals. On further analysis it was found that the *bear* is the only predatory mammal *without a tail* in this data set. The shape of the GSOM, by branching out, clearly brought the *bear* attention. In a more realistic data mining application, the user might identify interesting outliers with this type of analysis. The *mole* and the *hare* have been separated from their respective groups (nondomestic, nonpredatory mammals, and predatory mammals) due to their smaller size.

Such analysis can also be carried out on other clusters. The fish has been separated from other animals that were clustered close to fish. The reasons for such closeness would become apparent with further analysis of the new spread out GSOM. Since the data are already well separated, it would not be useful to further spread out the current clusters. But in a more complex data set, it might be necessary to use several levels of GSOM's with increasing SF values to gradually obtain a better understanding of the data set.

### C. The GSOM for High-Dimensional Human Genetic Data Set

In this section, a GSOM is generated for a more complex data set with 42 dimensions. The purpose of this experiment is to

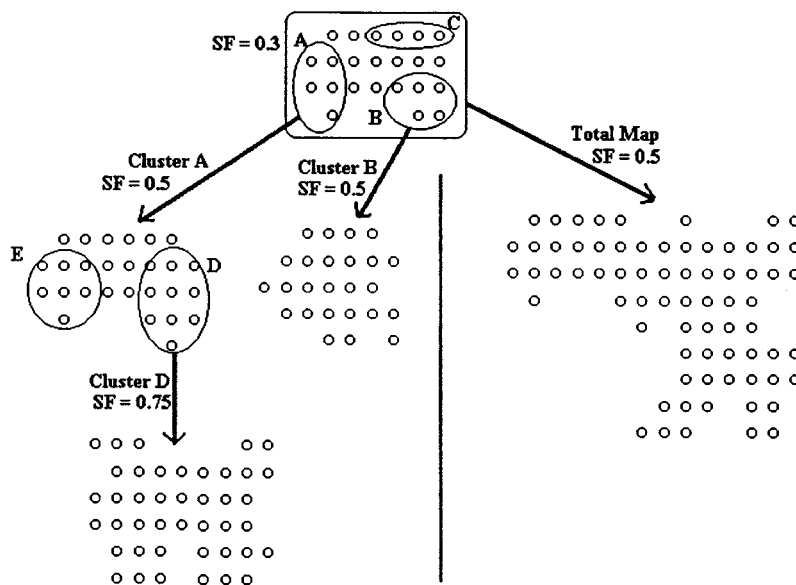


Fig. 10. The hierarchical clustering of a data set with increasing SF values.

TABLE I  
A RECORD FROM THE ZOO DATA SET

name	has hair	feathered	lay eggs	feeds milk	airborne	aquatic	predator	toothed	has backbone
Lion	1	0	0	1	0	0	1	1	1
name	breathes	venomous	has fins	no. of legs	has tail	domestic	is cat-size	type	
Lion	1	0	0	4	1	0	1	1	

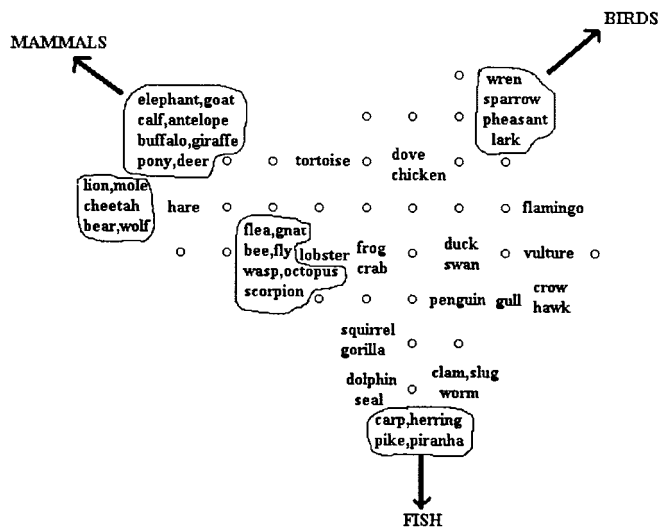


Fig. 11. The GSOM for the zoo data set with SF = 0.1.

demonstrate the ability of the GSOM to map a much higher dimensional and complex set of data. The human genetics data set consists of the genetic distances of 42 populations of the world, which have been calculated using gene frequencies [23]. The genetic information is derived from blood samples taken from individuals of a population inhabiting particular areas. The presence of certain genes has been identified in those blood samples, and these genes have been used to describe the individual. With sufficiently large samples, the individual gene frequencies have been used to calculate an average for each population. The data

set has been generated by selecting 42 populations from an initial 1950 populations. The *genetic distance* between the populations has been calculated with a measure called the  $F_{st}$ , which has specifically derived to measure distances between populations. The  $F_{st}$  calculation uses a form of normalization to account for frequencies that are not normally distributed. Special terms have also been added to correct any sampling errors. Thus  $F_{st}$  has been described as a better measure of genetic distance than the Euclidean distance [23].

By using SF = 0.5 to generate the GSOM for the genetics data, the result is shown in Fig. 14. It can be seen that the map is spread mostly according to the geographic localities. But some interesting deviations could be seen, such as the *Africans*' being separated into two subgroups. Fig. 15 shows a section of the genetics data further spread out for better understanding. The *left branch* of the GSOM in Fig. 14 was picked since it showed a *clustering* of populations, which were generally thought to be *different*. The selected populations were mapped with  $sf = 0.6$ , as shown in Fig. 15. The subgroups inside the cluster become apparent with the higher level of spread.

VII. DISCUSSION

The GSOM presented in this paper uses the basic concepts of self-organization as the SOM but has a dynamic structure that is generated during the training process itself. The main difference between the two methods is that the SOM attempts to *fit* in a data set into a predefined structure by *self-organizing* its node weights as well as possible within its fixed *borders*. With

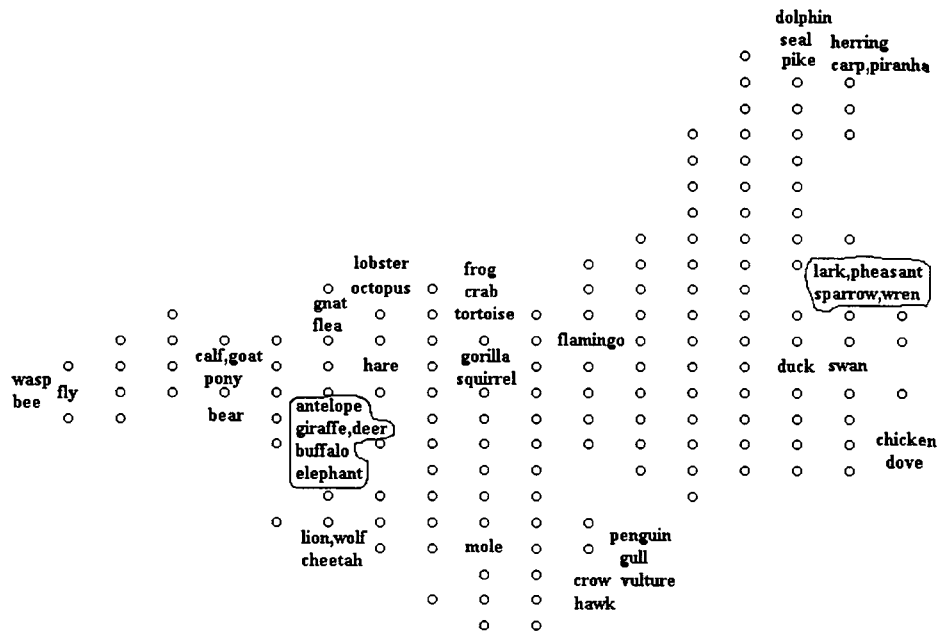


Fig. 12. The GSOM for the zoo data set with SF = 0.85.

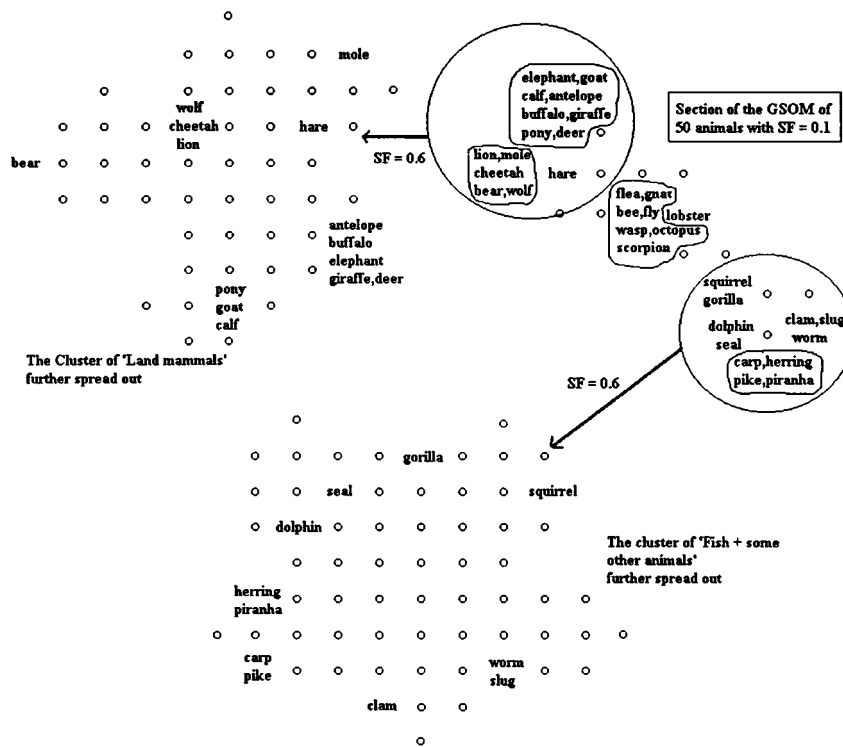


Fig. 13. The mammals cluster spread out with SF = 0.6.

the GSOM, the borders of the network are expandable, and as such the data set can generate new nodes with a *flowing out* effect, expanding the network outwards. Therefore, the different groupings in the data generate regions for themselves in the network. The self-organization of the weights in the already generated nodes continue at the same time to fine-tune the weights to represent the data better. This process generates the GSOM, which develops into different shapes depending on the clusters present in the data.

The advantages of the GSOM specially for knowledge discovery applications are as follows.

- 1) The GSOM can be used on a set of data, about which no previous information is available, to identify the clusters present (unsupervised method).
- 2) The shape of the GSOM represents the grouping in the data, and therefore, such grouping has a better opportunity of attracting the attention of the data analyst for further investigation.

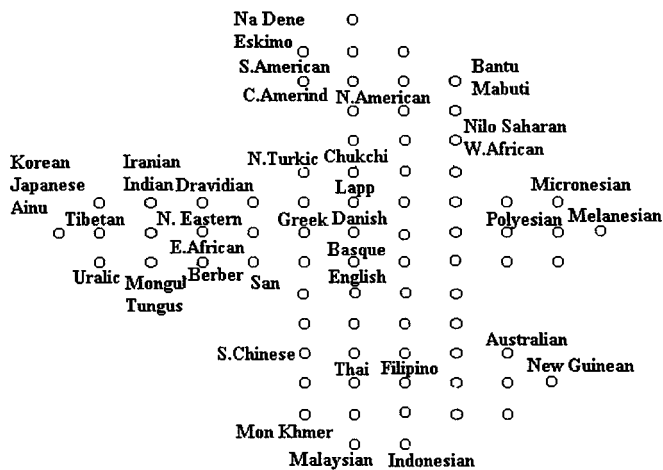


Fig. 14. The map of the human genetics data.

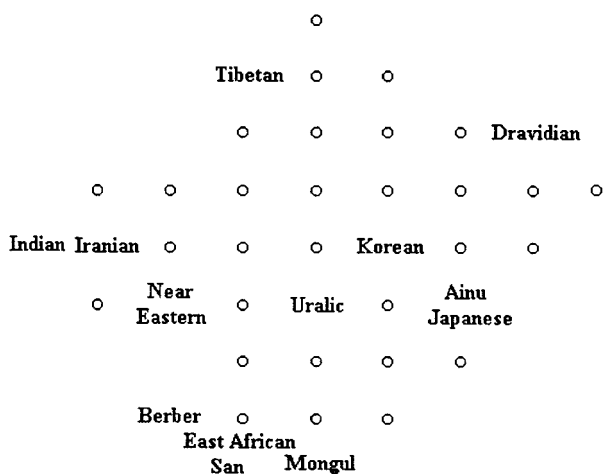


Fig. 15. Further expansion of the genetics data.

- 3) The number of nodes required to represent a set of data (at the same level of spread) is less than that of a comparable SOM. This will result in faster processing and efficient use of computing resources. Such comparisons with the traditional SOM have been described in [10], [11].
- 4) The weights of the newly generated nodes are initialized to fit in with the existing (current) map. This is possible due to the incremental addition of nodes to the network at the areas of necessity. In the SOM, the nodes are normally randomly initialized and have to be first *ordered*. The GSOM does not require an ordering phase, which further reduces processing time. The *ordered* weight initialization method also results in less chance of *twisted* maps.
- 5) Having fewer nodes at the early stages and localized weight adaptation also results in less processing time.

Advantages 3)–5) become very useful when dealing with large data sets. The SOM is currently being used for commercial data mining applications [1], [24]. The size of the network becomes a significant issue in such applications, as it will directly affect the processing time (not only due to the need of processing a larger number of nodes but also because the neighborhood sizes for weight adjustment would become larger). Due to its flexible

structure, the GSOM achieves the same amount of spread with a lesser number of nodes, and as such will provide a useful advantage in mapping large data sets. In addition, such flexible structure provides a better visualization of the groups in the data and attracts attention to such groups (and outliers) by branching out. Therefore, it is evident that the GSOM would be more useful as a clustering tool in knowledge discovery tasks.

In addition to these advantages, a method of controlling the GSOM is introduced using a spread factor. The ability to control the spread of the map will result in the data analyst’s having more control on the tests carried out. The data analyst can therefore generate a small map at the beginning (which would be faster and would take up less resources) and then select the *regions* for further analysis hierarchically. This would be very advantageous in handling larger data sets, as the processing can be focused on the areas of interest. Therefore, it should be added that the GSOM has preserved the simplicity and ease of use of the SOM and has expanded its usefulness by dynamically generating the structure. The new method of weight initialization and adaptation results in easier smoothing out, and the spread factor provides more control for the user.

REFERENCES

- [1] M. J. A. Berry and G. Linoff, *Data Mining Techniques*. New York: Wiley, 1997.
- [2] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering Data Mining —From Concept to Implementation*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
- [3] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag, 1995.
- [4] T. Villmann, R. Der, M. Hermann, and M. Martinetz, “Topology preservation in self-organizing feature maps: Exact definition and measurement,” *IEEE Trans. Neural Networks*, vol. 8, pp. 256–266, 1997.
- [5] H. Ritter, T. M. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps*. Reading, MA: Addison-Wesley, 1992.
- [6] S. Sestito and T. S. Dillon, *Automated Knowledge Acquisition*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
- [7] P. T. Quinlan, “Structural change and development in real and artificial neural networks,” *Neural Networks*, vol. 11, pp. 577–599, 1998.
- [8] T. Ash and G. W. Cottrell, “A review of learning algorithms that modify network topologies,” Univ. of California, San Diego, CA, Tech. Rep., 1994.
- [9] L. D. Alahakoon and S. K. Halgamuge, “Knowledge discovery with supervised and unsupervised self evolving neural networks,” in *Proc. Int. Conf. Information-Intelligent Systems*, 1998, pp. 907–910.
- [10] L. D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, “A structure adapting feature map for optimal cluster representation,” in *Proc. Int. Conf. Neural Information Processing*, 1998, pp. 809–812.
- [11] —, “A self growing cluster development approach to data mining,” in *Proc. IEEE Conf. Systems, Man, and Cybernetics*, 1998, pp. 2901–2906.
- [12] S. K. Halgamuge and M. Glesner, “Fuzzy neural networks: Between functional equivalence and applicability,” *Int. J. Neural Syst.*, 1995.
- [13] S. K. Halgamuge, “Self evolving neural networks for rule based data processing,” *IEEE Trans. Signal Processing*, vol. 44, no. 11, 1997.
- [14] B. Fritzke, *Let it Grow—Self Organizing Feature Maps with Problem Dependent Cell Structure*. Amsterdam, The Netherlands, 1991, pp. 403–408.
- [15] —, “Growing cell structure: A self organizing network for supervised and un-supervised learning,” *Neural Networks*, vol. 7, pp. 1441–1460, 1994.
- [16] T. M. Martinetz and K. J. Schulten, *A Neural Gas Network Learns Topologies*. Amsterdam, The Netherlands: Elsevier, 1991, pp. 397–402.
- [17] J. Blackmore, “Visualising high dimensional structure with the incremental grid growing neural network,” M.S. thesis, Univ. Texas at Austin, 1995.
- [18] T. Lee, *Structure Level Adaptation for Artificial Neural Networks*. Norwell, MA: Kluwer, 1991.

- [19] S. Jockush, *A Neural Network Which Adapts its Structure to a Given Set of Patterns*. Amsterdam, The Netherlands: Elsevier, 1990, pp. 169–172.
- [20] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams*. New York: Wiley, 1992.
- [21] E. Keogh, C. Blake, and C. J. Merz, “UCI repository of machine learning databases, 1999,”.
- [22] L. D. Alahakoon and S. K. Halgamuge, “Data mining with self evolving neural networks,” in *Advanced Signal Processing Technology*. New York: World Scientific, 2000, ch. 10.
- [23] L. L. Cavalli-Sforza, P. Menozzi, and A. Piazza, *The History and Geography of Human Genes*. Princeton, NJ: Princeton Univ. Press, 1994.
- [24] J. P. Bigus, *Data Mining with Neural Networks*. New York: McGraw-Hill, 1996.



Monash University.

**Dammina Alahakoon** received the B.Sc. (Hons.) degree in computer science from the University of Colombo, Sri Lanka, in 1994. He is currently pursuing the Ph.D. degree in data mining with artificial neural networks at Monash University, Australia.

From 1987 to 1990, he was a Credit Officer and an Accountant at Mercantile Credit Ltd., Sri Lanka. From 1994 to 1996, he was an Executive at the Colombo Stock Exchange, Sri Lanka. Since 1996, he has also been an Assistant Lecturer at the School of Computer Science and Software Engineering,



**Saman K. Halgamuge** (M'85) received the B.Sc. degree in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 1985 and the Dipl.-Ing. and Dr.-Ing. degrees in computer engineering from Darmstadt University of Technology, Germany, in 1990 and 1995, respectively.

In 1985, he was an Engineer at the Ceylon Electricity Board, Sri Lanka. From 1990 to 1995, he was a Research Associate at Darmstadt University of Technology. After lecturing in computer systems engineering, he was associated with the Institute for Telecommunications Research and the School of Physics and Electronic Engineering Systems of the University of South Australia, Australia. Since 1996 he has been with the Department of Mechanical and Manufacturing Engineering, University of Melbourne, Australia, as a Senior Lecturer in mechatronics. He is a coauthor of about 85 conference/journal papers and has contributed to books in the areas of data mining and analysis, mechatronics, neural networks, genetic algorithms, and fuzzy systems. His research interest also includes data and information fusion, communication networks, and modeling and simulation in manufacturing systems.

**Bala Srinivasan** received the B.Eng. (Hons.) degree in electronics and communication engineering and the master's and Ph.D. degrees in computer science, all from the Indian Institute of technology, Kanpur, India.

He is a Professor of information technology in the School of Computer Science and Software Engineering, Faculty of Information Technology, Monash University, Melbourne, Australia. He was formerly an Academic Staff Member of the Department of Computer Science and Information Systems, National University of Singapore, and the Indian Institute of Technology, Kanpur, India. He has authored and jointly edited six technical books and is an author or coauthor of more than 150 international refereed publications in journals and conferences in the areas of multimedia data bases, data communications, data mining, and distributed systems. He is a Founding Chairman of the Australasian Database Conference.