Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map

Justine Blackmore and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin, Austin, TX 78712-1188
justine,risto@cs.utexas.edu

Technical Report AI92-192
December 1992

Abstract

Knowledge of clusters and their relations is important in understanding high-dimensional input data with unknown distribution. Ordinary feature maps with fully connected, fixed grid topology cannot properly reflect the structure of clusters in the input space—there are no cluster boundaries on the map. Incremental feature map algorithms, where nodes and connections are added to or deleted from the map according to the input distribution, can overcome this problem. However, so far such algorithms have been limited to maps that can be drawn in 2-D only in the case of 2-dimensional input space. In the approach proposed in this paper, nodes are added incrementally to a regular, 2-dimensional grid, which is drawable at all times, irrespective of the dimensionality of the input space. The process results in a map that explicitly represents the cluster structure of the high-dimensional input.

1 Introduction

The self-organizing feature map's (Kohonen 1989, 1990) primary use as a computational tool is to form a mapping from a high-dimensional input space to two dimensions. How useful the map is for a given task depends on how accurately it represents the input space. In general, the input space may be arbitrarily nonconvex and discontinuous, and may contain high-dimensional clusters. A good representation of the space should somehow capture such topological properties. However, accurately representing high-dimensional structure on a continuous, fully connected $n \times m$ grid is problematic. Discontinuities in the input space may appear bridged in the map. The map may have connections that span the disjoint clusters, or it may have nodes situated within the discontinuity (where the input probability is 0; figure 1). In other words, the final feature map sometimes misrepresents the topology of the input data.

Real world data sets often contain distinct but non-obvious subsets of data. Determining the set of classifications that optimally describes such subgroupings is a primary goal of many standard "clustering" methods. Because of bridging the gaps, the standard self-organizing algorithm does not naturally delineate the boundaries of such groupings. A feature map application that depends on an accurate representation of neighborhood boundaries would need to perform further analysis to determine if discontinuities have been inaccurately spanned in the map.

This paper describes an incremental grid-growing algorithm for incorporating such information directly into the structure of the map. During organization, non-convexities, discontinuities and clusters in the data set become explicitly represented in the 2-dimensional structure of the map. Thus the algorithm can yield an accurate, low-dimensional description of the structure in high-dimensional input.

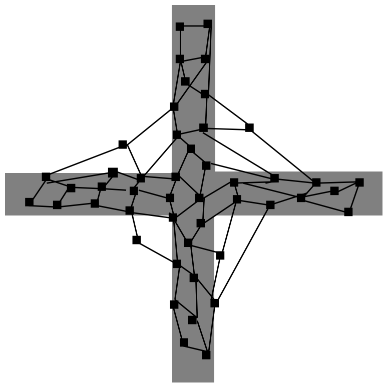


Figure 1: **Representing nonconvex input distributions with ordinary feature maps.** The input space consists of 2-dimensional vectors uniformly distributed in the shaded region. The black dots indicate locations of the feature map weight vectors after the map has been organized. Weight vectors of neighboring nodes in the map are connected with lines. Note that nodes are allocated to areas where there is no input. Similarly, connections sometimes span these areas, suggesting that the connected nodes represent neighboring vectors in the input space although in reality they belong to different clusters.

2 An incremental approach

In order to develop an accurate representation of the topology, the self-organizing algorithm must either recognize and correct misrepresentations that develop in the map, or else prevent such incorrect topology from being encoded in the first place. Completely preventing the development of inaccurate structure is impossible without a priori knowledge of the input space. On the other hand, fully organizing a map and then modifying it so that unwanted structures are removed may require much extra computational effort. In general, an algorithm must be equipped with some effective heuristics to accomplish both ends: to guide the development of structure actually present in the data set, and to detect and correct any "false" topology in the map as early as possible during organization.

These considerations suggest an incremental approach to building and organizing the map. Such an approach would initially organize a small number of nodes in the structure, then use a heuristic to find and remove any potentially inaccurate nodes or connections. Another heuristic could be used to add nodes to the structure. The new structure would be re-organized, and the process continued. At every epoch, the algorithm would guide the map toward representing the high-dimensional properties of the data set accurately.

Fritzke's (1991a, 1991b) growing cell structure algorithm is one such incremental approach. The basic structure of the map, however, is not a 2-dimensional grid of nodes, but rather a structure whose connections at all times define a system of triangles (i.e., every node must always be a member of a triangle). During organization, a heuristic measure is used to determine areas of the structure that inadequately represent their corresponding areas of the input space, and new nodes are added in these areas. Also, nodes that rarely respond maximally to the input are periodically removed from the structure.

The algorithm results in a network structure that represents an arbitrarily connected graph $G = (V, E)$, where V is the set of nodes, and E is the set of connections between them. In the case of 2-dimensional input, it is easy to verify that the network structure accurately represents the input by plotting the weight vectors in 2-D. When the input is high-dimensional, however, such an arbitrary structure may not have a simple low-dimensional description (that is, it cannot easily be drawn in 2 dimensions). Although guaranteed to be planar, the graph is not guaranteed to have a regular or symmetric planar embedding; indeed, this is very unlikely. Also, the arbitrary connectivity makes topological neighborhoods ambiguous beyond directly connected nodes. Any node may be connected to any number of neighbors, so a neighborhood of a given radius in the structure (i.e., the number of connections outward in any direction) has little topological meaning. Thus, extracting the overall topological relationships of the input space from this structure may not be easy. The algorithm does explicitly represent clustering of the input data by removing connections between the clusters in the structure. However, the topology within clusters and across continuous data sets may be difficult to determine.

The incremental grid-growing algorithm described in this paper is also based on the incremental approach, but it avoids the difficulties of an arbitrarily connected graph structure. The map retains a regular 2-dimensional grid structure at all times. At any point during the organization, the map has a simple 2-dimensional description, and topological relations are easily examined by plotting the nodes and connections present in the map in 2-D.

3 The grid-growing algorithm

Initially, the feature map grid consists of four connected nodes with random weight vectors (figure 2a). Each main iteration of the algorithm consists of three steps:

1. Adapting the current grid to the input distribution through the usual feature map self-organizing process.
2. Adding nodes to those areas in the perimeter of the grid that inadequately represent their corresponding input area.
3. Examining the weight vectors of neighboring nodes and determining whether a connection between the nodes should be deleted from the map, or whether a new connection should be added.

The new structure is re-organized, and the process continues until a predetermined maximum number of nodes has been reached.

Each step will now be described in more detail. A boundary node is defined as any node in the grid that has at least one directly neighboring position in the 2-dimensional grid space not yet occupied by a node. Each boundary node maintains an error value E. During each iteration, whenever such a node wins an input presentation (i.e., the Euclidean distance between the node's weight vector and the input vector is minimum for the map), the square of the distance between its weight vector and the input is added to the error value:

$$E(t) = E(t-1) + \sum_k (x_k - w_k)^2, \tag{1}$$

where E is the cumulative error and w the weight vector of the winning unit, and x is the input vector.

At the end of each iteration, the boundary node with the greatest cumulative error can be said to represent the area of the input space most inadequately represented in the map. This node "grows" new nodes in all unoccupied grid locations in its immediate neighborhood (figures 2a-d). New nodes are directly connected to the error node. If any other directly neighboring grid spots are occupied (as in figure 2d), the new node's weight vector is initialized to be the average value of all the neighboring weight vectors:

$$w_{NEW,k} = 1/n \sum_{i \in N} w_{i,k}, \tag{2}$$

where $w_{NEW,k}$ is the kth component of the new unit's weight vector and N is the set of the n neighboring nodes of the new unit. Otherwise (as in figure 2b), the new node's weight vector is

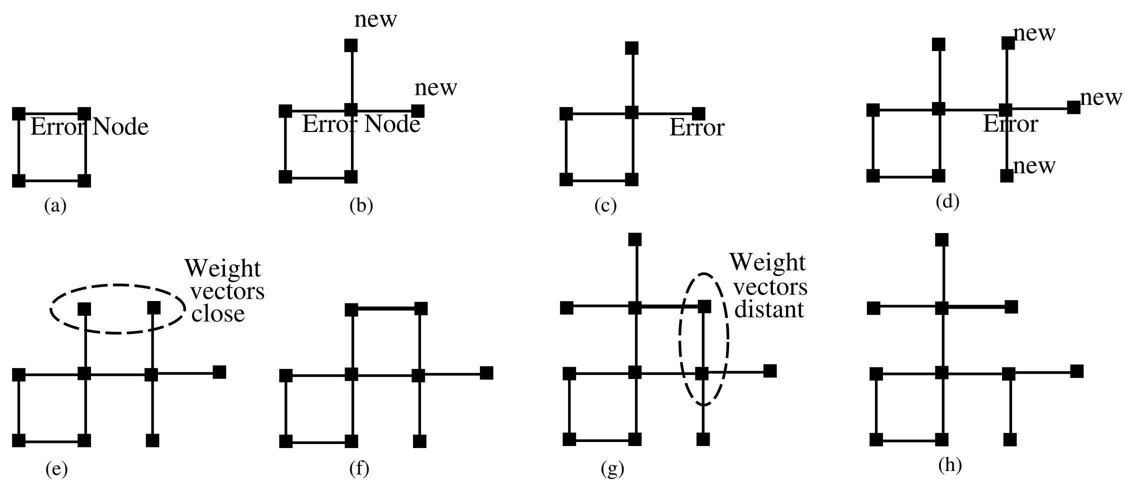


Figure 2: **Growing the feature map grid.** Figure (a) shows the initial structure after the first organization stage; the boundary node with the highest error value is marked. (b) New nodes are "grown" into any open grid location that is an immediate neighbor of the error node. (c) After organizing the new structure, a new error node is found. (d) Again, new nodes are grown into any open grid location that is an immediate neighbor of the error node. (e) During self-organization of this new structure, the algorithm detects that the circled nodes have developed weight vectors very close in Euclidean distance. (f) These "close" nodes are connected. (g) After further organization, the algorithm discovers connected neighboring nodes whose weight vectors occupy distant areas of the input (i.e., the nodes have a large Euclidean distance). (h) These "distant" nodes are then disconnected in the grid.

initialized so that the weight vector of the error node is the average of the new node's vector and the vectors of any already existing neighbors of the error node:

$$w_{ERR,k} = 1/(m+1)\left(w_{NEW,k} + \sum_{i \in M} w_{i,k}\right), \tag{3}$$

where $w_{ERR,k}$ is the kth component of the error node's weight vector and M is the set of the m already existing neighbor units of the error node.

Initially, the new nodes are connected to the structure only through the error node. As the structure continues to organize, these new nodes may develop weight vectors that are close to the weight vectors of neighbors to which they have not been connected. If this is the case, it is desirable to add a new connection joining these nodes. An adjustable threshold parameter is used to decide if a new connection should be grown. After each organizational iteration, the Euclidean distance between unconnected neighboring nodes in the map is examined. If the distance is below the "connect" threshold parameter, a connection between the nodes is added to the structure (figures 2e-f).

Similarly, a "disconnect" threshold parameter is used to determine if there are two nodes in the map that are connected even though they represent points that are distant in the input space. Exceeding such a threshold may indicate that a connection spans a discontinuity in the input, and should be removed from the map (figures 2g-h).

If the input distribution forms a connected area (as do the four arms of the cross in figure 1), in practice it is rarely necessary to delete connections from the grid. But it is possible to have disjoint input clusters in the data set—for example, the four arms might be separated by gaps. In general, Euclidean distances between clusters are greater than the intra-cluster distances. The clusters become separated from each other when the connections that span the inter-cluster gaps are removed by the algorithm. If the disconnect threshold is selected properly, distinct clusters will become separated in the map. The portions of the grid representing the independent clusters will continue to develop according to the topologies of the individual data clusters.

Adding nodes only at the perimeter allows the map to develop an arbitrary topology. Further, adding nodes only to areas that inadequately represent the input encourages the map to develop only those topological structures that are actually present in the data. Disconnecting nodes that span an apparent discontinuity allows clusters to separate and to continue to develop independently. The clusters that automatically develop may represent categories or sub-sets within the data set. Capturing such properties of the input space in the 2-dimensional structure of the map can greatly assist the interpretation of the input data.

4 Examples

The topology of an arbitrary high-dimensional space is difficult to visualize without a dimensionality-reducing tool such as a feature map. On the other hand, the topology of a 2-dimensional data set is trivial to visualize. Thus, for illustrative purposes, an experiment where the incremental grid-growing algorithm developed a feature map of a 2-dimensional input space is presented in figure 3. As in figure 1, 2-dimensional vectors were chosen with uniform probability from the cross-shaped area. The grid developed four arms connected through an area that represents the central portion of the cross. Each arm is represented by approximately the same number of nodes, and the central region is represented by a proportionately lower number of nodes. The clusters in this input space are joined in the central region; this structure is duly reflected by the resulting grid. Note that the grid structure itself, even without any labelling of nodes, follows the overall topology of the input space. The structure of the data set and its overall probability density are encoded in the structure of the map.

Let us now extend this example to higher-dimensional input. Consider the case where each of the arms is a 4-dimensional "box" along one of 4 coordinate axes. Three of these arms are connected in a 4-dimensional area surrounding the origin, while the fourth region is separated from the origin by a gap. That is, the input vectors (w, x, y, z) are uniformly distributed within the 4-dimensional area defined by the union of these 4 areas:

$$\begin{aligned}
\text{region } 1 &= \{(w,x,y,z) : 0 \leq w < 5; 0 \leq x < 1; 0 \leq y < 1; 0 \leq z < 1\} \\
\text{region } 2 &= \{(w,x,y,z) : 0 \leq w < 1; 0 \leq x < 5; 0 \leq y < 1; 0 \leq z < 1\} \\
\text{region } 3 &= \{(w,x,y,z) : 0 \leq w < 1; 0 \leq x < 1; 0 \leq y < 5; 0 \leq z < 1\} \\
\text{region } 4 &= \{(w,x,y,z) : 0 \leq w < 1; 0 \leq x < 1; 0 \leq y < 1; 2 \leq z < 6\}
\end{aligned}$$

The final map representing this structure is shown in figure 4. As in the 2-dimensional case above, the first three regions are connected through a central region, and 3 arms extend outward. The fourth region is fully separated from the first three. The relative numbers of nodes throughout the structure reflect the uniform distribution of the input. Again, the overall topology and distribution of the input space is apparent in the simple 2-dimensional structure of the grid.

The final example, the "spanning tree" of Kohonen (1990), demonstrates the algorithm's ability to develop arbitrarily complex topologies for discrete inputs. In this example, the input consists of the 5-dimensional vectors listed in figure 5a. The high-dimensional topology of this data set is difficult to describe, but a minimum spanning tree is one possible structure that could be used to represent it (figure 5b). Indeed, if one knows what to look for, it is easy to see that the map developed by Kohonen with the standard self-organizing algorithm displays the spanning tree structure (figure 6a). However, the full connectivity of the map makes it difficult to discern a true tree or graph structure from the map alone.

The incremental grid-growing algorithm applied to the same data set derives a map that makes the arrangement much clearer (figure 6b). The arms of the spanning tree are clustered in delineated regions of the map. Also, the relationships between the clusters are narrowly specified by the limited

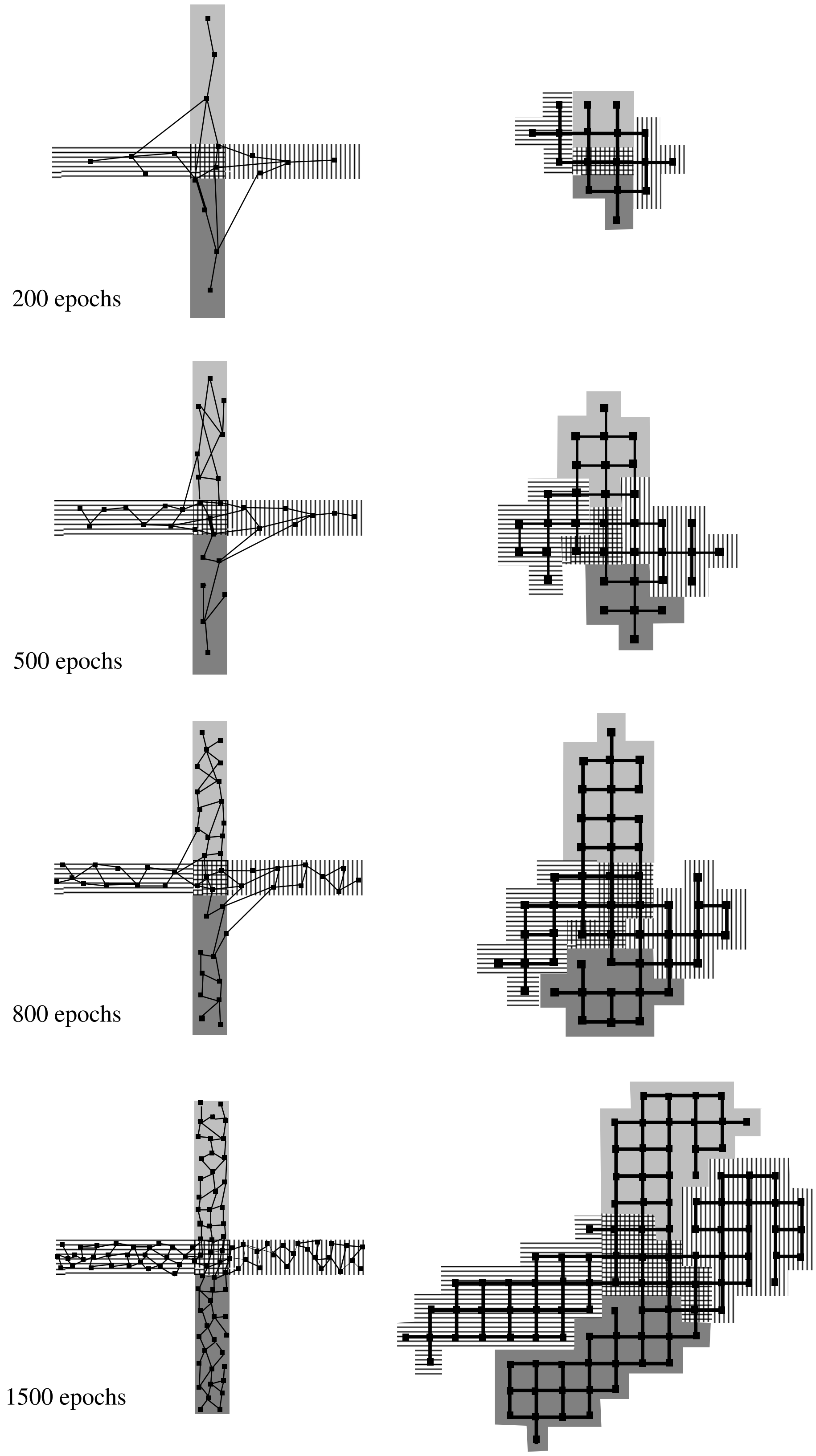


Figure 3: **Snapshots of the grid evolution for a 2-dimensional cross input.** On the left, the weight vectors are plotted on the 2-d input space. On the right, the corresponding grid structure is shown. Shading of the area around a node indicates the arm of the cross where that node's weight vector is located. The four arms are separated in the grid, with a common center.

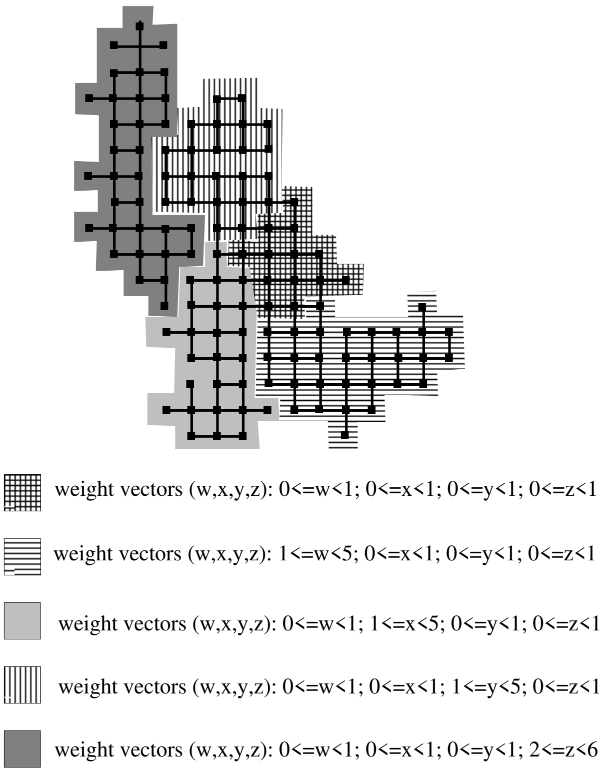
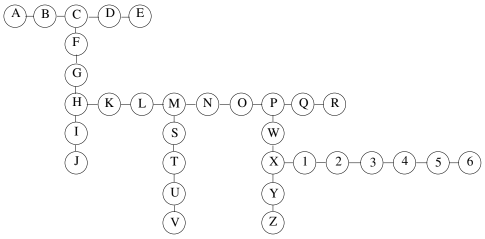


Figure 4: **Final grid structure for 4-dimensional, disjoint input space.** The common center (the first area) and the first three arms are connected, while the fourth arm is fully separated.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

(a)



(b)

Figure 5: **(a) The input set for Kohonen's (1990) spanning tree example. (b) The minimal spanning tree of the data.** Although the example may seem artificial, it illustrates the self-organizing feature map's capacity to represent the general topology of difficult-to-describe data sets. The data has no obvious or easily discernible description, but the minimal spanning tree is one relational description that might be derived.

connectivity between them. The structure of the map obviates the need for a priori knowledge of the spanning tree. Interestingly, "conventional" clustering methods (e.g., merge clustering) do not naturally derive such graph-like relationships in this type of data.

5 Conclusion and future work

The incremental grid-growing algorithm constructs 2-D drawable feature maps of arbitrary non-convex and discontinuous high-dimensional input distributions. The algorithm addresses a primary shortcoming of the standard SOFM: deciding where the boundaries of clusters and specific regions are on the map. The overall topology of the input space is encoded in the network structure alone, before any node labelling has been done. Because the 2-dimensional clusters that develop can be

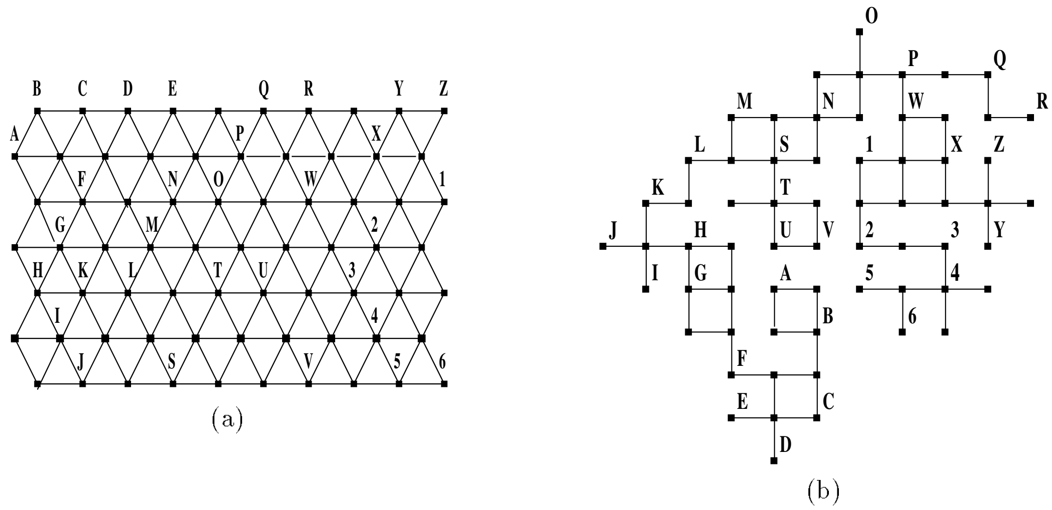


Figure 6: **Feature map representation for the spanning tree data.** (a) Map derived by the standard self-organizing algorithm (Kohonen 1990). The map is hexagonally connected. The spanning tree structure is clearly present in the map; however, the full connectivity makes it difficult to extract exact neighborhood relations between units. (b) Map derived by the grid growing algorithm. The limited connectivity between clusters in the map closely resembles the structure of the spanning tree.

interpreted as categories, the grid is a useful tool for data classification. Furthermore, it can extract and represent tree- or graph-like structures, which makes it useful in visualizing the relationships in complex high-dimensional data sets.

Our ongoing work on grid growing includes fine-tuning the basic algorithm and applying it to real-world problems. Like any other feature map algorithm, the incremental grid algorithm could be improved by a method for setting the threshold values automatically (removing the need for parameter tuning). It would also be desirable to develop a computational measure that could be used decide when a map has developed a good representation for an arbitrary data set.

In addition, the application of the algorithm to various complex high-dimensional data sets is being investigated. For example, the algorithm should be useful in developing clusters for the representation of semantic features of words (e.g. Miikkulainen and Dyer 1991; Ritter and Kohonen 1989). Data interpretation and knowledge representation in general is a most promising application of feature map algorithms, and incremental grid growing should prove particularly useful in such tasks.

References

Fritzke, B. (1991a). Let it grow—Self-organizing feature maps with problem dependent cell structure. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 403–408. Amsterdam; New York: North-Holland.

Fritzke, B. (1991b). Unsupervised clustering with growing cell structures. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), vol. II, 531–536. Piscataway, NJ: IEEE.

Kohonen, T. (1989). *Self-Organization and Associative Memory.* Berlin: Springer. Third edition.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480.

Miikkulainen, R., and Dyer, M. G. (1991). Natural language processing with modular neural networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Ritter, H. J., and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 61:241–254.